3D Reconstruction and Nearest 6-DoF Pose Estimation using Truncated Signed Distance Functions

Gaurav Behera
IIIT Hyderabad
2022111004
gaurav.behera@research.iiit.ac.in

Samkit Jain
IIIT Hyderabad
2022102062
samkit.jain@students.iiit.ac.in

Ansh Chablani
IIIT Hyderabad
2022111031
ansh.chablani@research.iiit.ac.in

***Abstract*—In this paper, we present a pipeline for real-time 3D reconstruction and nearest 6 Degrees of Freedom (6DOF) pose estimation of objects from monocular RGB-D video data. Our approach integrates multiple computer vision and robotics techniques to provide the nearest pose estimation, giving a good initialization for more robust 6DOF pose estimation methods that can be used for applications such as augmented reality, robotics, and autonomous systems. The pipeline is composed of three main steps: (1) construction of a dense 3D model using Truncated Signed Distance Function (TSDF) fusion from consecutive depth maps, (2) estimation of the object's 6DOF pose, and (3) generation of a high-resolution 3D mesh through the Marching Cubes algorithm and Occupancy Dual Contouring algorithm, which allows for refined pose estimation via differential rendering. Our approach aims to provides accurate, efficient, and scalable solutions for 3D object tracking and pose estimation in real-time applications.**

GitHub Repository: https://github.com/anshium/mr-project

I. Introduction

In the 6 Degrees of Freedom (6DOF) Pose Estimation problem, the objective is to accurately determine the position and orientation of an object within a three-dimensional space, often using RGB or RGB-D images. The term 6DOF refers to the six essential parameters required to describe the object's pose: three for translation and three for rotation with respect to the camera's reference frame. Accurate prediction of these parameters is crucial for enabling a robot or computer vision system to perceive the exact location and orientation of an object relative to the camera or within a global coordinate system. This understanding of an object's spatial properties is fundamental to various applications, such as robotic manipulation, augmented reality, and autonomous navigation.

II. Related Work

A. CAD Model Based Pose Estimation

Older methods for 6DOF pose estimation relied on CAD models of objects as ground-truth 3D references. These approaches typically followed a sequence: object detection, feature extraction, feature matching to the CAD model, and pose refinement.

- **Object Detection and Segmentation:** The object of interest is detected using handcrafted features, such as edges, contours, and keypoints (e.g., SIFT or ORB). In some cases, color or texture information is used to segment the object from the background, and CAD models for multiple objects are stored for use in the pipeline.
- **Feature Extraction and Matching:** After detection, visual features from the image are matched to predefined keypoints on the CAD model using descriptors like SIFT or SURF, establishing 2D-3D correspondences.
- **Pose Estimation Using PnP:** The Perspective-n-Point (PnP) algorithm estimates the 6DOF pose by aligning 2D image points with corresponding 3D CAD points, using intrinsic camera parameters for accurate mapping.
- **Pose Refinement:** Optimization methods like ICP refine the initial pose by minimizing alignment errors between the CAD model and the observed image.

The limitations of CAD-based methods include sensitivity to occlusions, lighting, and texture variations, making feature matching challenging for low-texture or repetitive objects. Each object requires a unique CAD model and keypoint annotation, making these methods difficult to scale. Reflective surfaces, such as glass or metal, create specular highlights and reflections that introduce false keypoints, leading to inaccurate pose estimates. Classical methods struggle with these challenges, as reflections create variable, viewpoint-dependent patterns that interfere with consistent feature matching.

B. CAD Model-Free Approaches (OnePose [4] and OnePose++ [2])

While CAD Model based approaches have a much higher accuracy, over the years several CAD Model-Free Approaches have tried to match the accuracy of these model. Here we discuss the more recent ones.

OnePose++ proposes a keypoint-free one-shot object pose estimation method, called OnePose++, that does not require CAD models. The method is inspired by the feature-matching-based method, OnePose, and uses a keypoint-free feature matching pipeline to reconstruct a semi-dense point-cloud model for an object from an image sequence with annotated poses. Given a query image, the method uses a 2D-3D matching network to directly establish correspondences between the

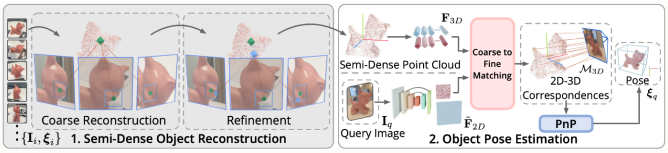


Fig. 1: OnePose++ Pipeline Overview

query image and the reconstructed point-cloud model, without first detecting keypoints in the image.

OnePose++ is a **two-stage pipeline** that consists of a keypoint-free structure from motion (SfM) stage and an object pose estimation stage.

- **Stage 1: Keypoint-Free Structure from Motion Stage** This stage reconstructs the object's semi-dense point cloud from a reference image sequence with known object poses using a coarse-to-fine scheme that leverages the coarse and fine stages of the LoFTR [3] keypoint-free feature matching method.
- **Stage 2: Object Pose Estimation Stage** The object pose estimation stage takes a query image and establishes 2D-3D matches between the reconstructed object point cloud and the image to estimate the object's pose using a coarse-to-fine scheme for efficiency.
- **Evaluation** The method is evaluated on the OnePose, LINEMOD and OnePose-LowTexture datasets, using the cm-degree pose success rate, ADD(S)-0.1d, and 2D projection error metrics. The experiments show that OnePose++ claims to achieve state-of-the-art results compared to existing one-shot object pose estimation methods, and claims to achieve comparable results with instance-level methods that require object CAD models.

C. Bundle Adjustment Methods

BundleSDF [5] proposes a novel method for 6-DoF pose tracking and 3D reconstruction of unknown, dynamic objects from monocular RGBD video sequences. The method only requires a 2D object mask in the first frame of the video and assumes the object is rigid.

- **Coarse pose initialization:** BundleSDF obtains an initial estimate of the object pose in the current frame by matching features with the previous frame using an object-agnostic video segmentation network. The identified correspondences, along with depth information, are used by a RANSAC-based pose estimator [1] to determine the coarse pose estimate.
- **Memory Pool:** A key-frame memory pool is maintained to alleviate catastrophic forgetting and reduce long-term tracking drift. The first frame is used for defining the canonical coordinate system for the unknown object. Subsequent frames are added only if their viewpoint significantly enhances the multi-view diversity of the pool.
- **Pose Graph Optimization:** An online pose graph optimization process is used to refine the object pose estimates over time. A subset of frames from the memory pool is dynamically selected to participate in the optimization which jointly refines the poses of the selected frames and the current frame, minimizing a loss function that accounts for pairwise edge losses and a unary loss.
- **Neural Object Field:** A Neural Object Field is concurrently trained with the pose graph optimization to reconstruct the 3D shape and appearance of the object. The Neural Object Field uses a signed distance function (SDF) representation which learns to represent the object's geometry and appearance from the posed memory frames, refining the previously estimated poses

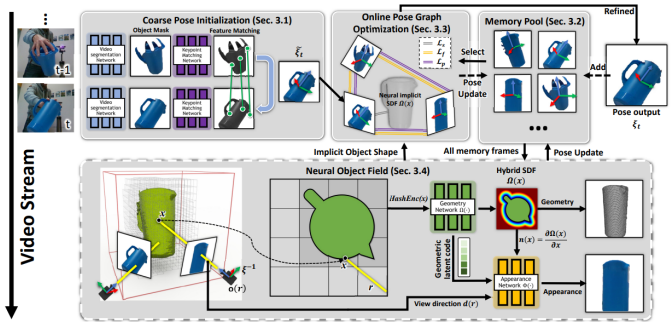


Fig. 2: BundleSDF Framework Overview

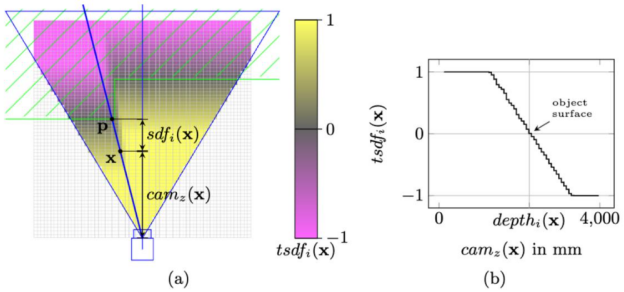


Fig. 3: 2D TSDF example. (a) Solid object (green), camera with field of view, optical axis and ray (blue), and TSDF grid (unseen voxels are white, for others see color bar). The signed distance value of voxel x is determined by the depth of the corresponding surface point p and the voxel's camera distance cam(x). (b) 1D TSDF sampled along the ray through p with t = 1000 mm. Object surface is at zero crossing.

BundleSDF achieves state-of-the-art performance on several benchmarks, including HO3D, YCBInEOAT, and BEHAVE datasets. It exhibits robustness in handling challenging scenarios, such as fast motion, occlusions, lack of texture, and specular highlights.

D. Truncated Signed Distance Function

Truncated Signed Distance Function [6] is a volumetric scene representation used for integrating multiple depth images taken from different viewpoints to reconstruct a 3D model of a scene. TSDF is favored for its efficiency in terms of both

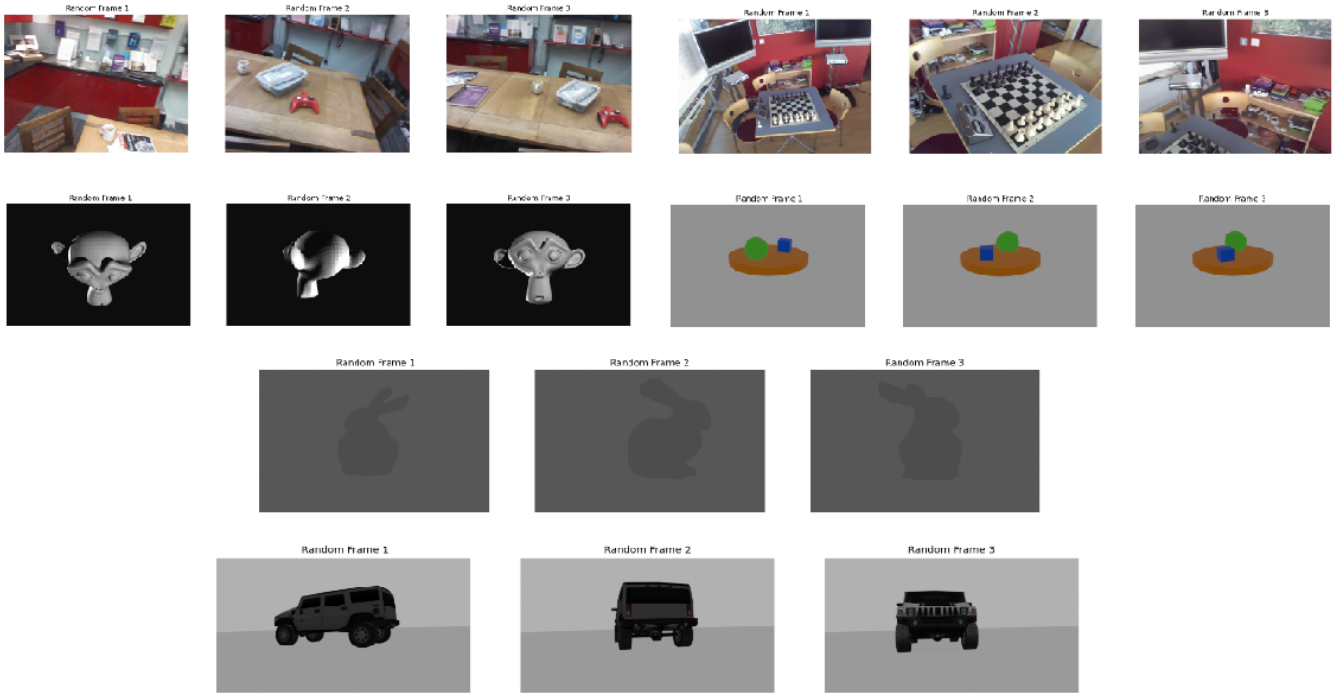


Fig. 4: Datasets

time and space, and its suitability for parallel processing on GPUs, allowing for real-time processing at high frame rates.

- **Representing the Scene:** The scene is broken into a 3D grid made up of equally sized cubes called voxels. Each voxel stores information about its distance to the nearest object surface.
- **Signed Distance Function (SDF):** For each voxel, the SDF value is calculated which is the distance between the center of the voxel and the closest point on the object's surface along the line of sight from the camera.
- **Truncation:** In TSDF, the SDF values are truncated at a certain distance (±t). This means any distances beyond this threshold are simply assigned the maximum or minimum truncated value. This truncation has two main benefits - memory efficiency and focus on relevant information.
- **Integrating Multiple Observations:** A key strength of TSDF is its ability to combine data from multiple depth images. Each new observation is integrated into the existing TSDF using a weighted average. This iterative updating process helps to refine the 3D model over time and fill in any gaps or missing information from individual viewpoints.
- **Surface Reconstruction:** To reconstruct the surface, a technique called ray casting is used. Rays are cast from the camera's viewpoint through the TSDF grid. When a ray crosses from a positive TSDF value to a negative one, it indicates an intersection with the object surface. The precise intersection point is then estimated using interpolation between the surrounding TSDF values.

III. Dataset

Three primary data sources were employed: the 7 Scenes dataset, commonly used for TSDF-based scene reconstruction, and synthetic datasets generated using Blender and Gazebo for controlled experimentation.

A. 7-Scenes Dataset

The 7-Scenes dataset is a widely used benchmark for scene reconstruction and pose estimation tasks. It consists of RGB-D video sequences captured in realistic indoor environments. The dataset includes 7 distinct scenes: Chess, Fire, Heads, Office, Pumpkin, RedKitchen, and Stairs. Each scene provides the RGB images and depth maps recorded with a Kinect sensor and the ground-truth 6DOF poses corresponding to each RGB-D frame.

B. Blender

We generated synthetic data using Blender to create controlled environments for testing and validating our pipeline. The dataset generation process involved several key steps to ensure high-quality RGB images, depth maps, and accurate camera poses. First, we designed 3D environments by importing object models, arranging them within the scene, and setting up textures and materials, light sources such as point lights, ambient lights for visual fidelity. Using Blender's animation tools, we created camera trajectories by defining keyframes for position and orientation, enabling smooth transitions and dynamic viewpoints across the scene. Depth maps were computed using Blender's Render Passes, where the Z-depth data from the scene was captured and scaled.

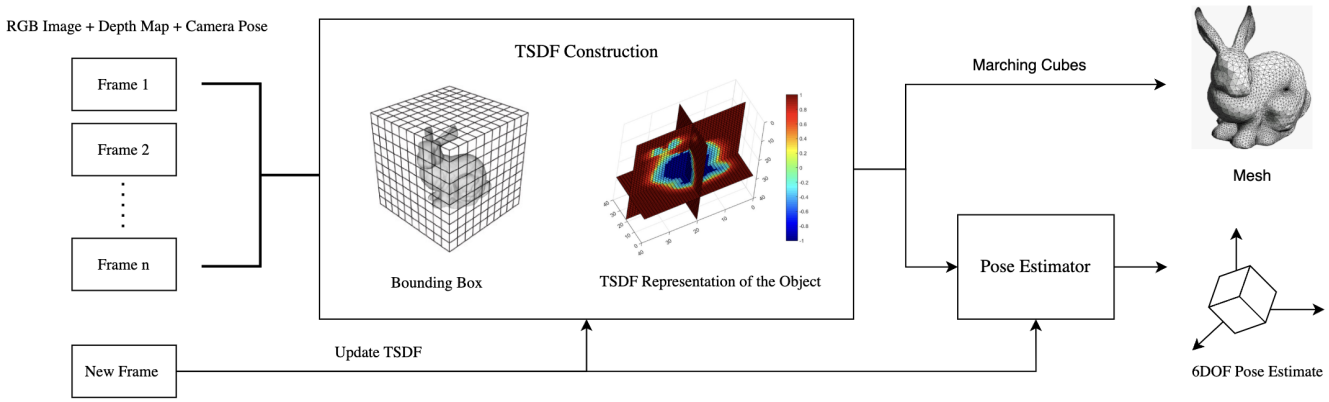


Fig. 5: Pipeline for real-time 3D reconstruction and pose estimation of objects from video data. The process consists of three main stages: (1) 3D model construction using Truncated Signed Distance Function (TSDF) fusion, which integrates RGB images, depth maps, and camera poses to build a volumetric representation of the object; (2) 6 Degrees of Freedom (6DOF) pose estimation to determine the object's position and orientation; and (3) mesh generation for refined pose estimation, producing a detailed 3D model suitable for applications in real-time tracking, augmented reality, and robotics.

To streamline the process, we developed Python scripts utilizing Blender's API to automate the rendering and data extraction pipeline. These scripts exported RGB images, normalized depth maps, and stored camera poses as transformation matrices. All data—RGB images, depth maps, and camera poses—were synchronized and saved in a structured format, ready for direct integration into our TSDF pipeline. The following datasets were generated using Blender: Suzanne (non-textured), Stanford Bunny (non-textured) and Table with objects (textured)

C. Gazebo

We also generated a dataset using a robot simulation done with ROS and Gazebo. This involved moving a robot with a depth camera around another object, which publishing the depth image, RGB image and the poses on separate topics and recording these topics in a rosbag.

Then we extracted the topics and converted to the format of the reconstruction script while making several transformations and resampling which was required due to different sampling rates of the images and the pose information of the camera on the robot.

A model SUV was generated using a differential mobile robot on which an Intel RealSense simulated camera was installed. The robot was controlled using a tele-operation method on the /cmd_vel topic.

IV. Method

We propose a comprehensive pipeline for the real-time 3D reconstruction and pose estimation of objects from video data. The method leverages RGB images, depth maps, and camera poses as inputs, transforming this data into a 3D model that captures the object's shape, orientation, and position. Our approach is designed to handle dynamic input from video, making it robust for applications in real-time tracking, augmented reality, robotics, and virtual environments. The process is divided into three main steps: constructing a 3D model using Truncated Signed Distance Function (TSDF) fusion, estimating the 6 Degrees of Freedom (6DOF) pose, and generating a 3D mesh for refined pose estimation. Below, we delve into each step with a detailed description.

A. TSDF Construction for 3D Reconstruction

The Truncated Signed Distance Function (TSDF) is a volumetric representation where each voxel encodes the signed distance to the nearest surface. The proposed pipeline is designed to be both efficient and adaptable for CPU and GPU processing, allowing scalability across various applications.

1) **Volume Bounds**
   Using the depth maps and camera parameters, the spatial bounds of the scene are computed. First, each pixel in the depth map is projected into the 3D world in the camera frame using the inverse of the camera intrinsics. These points are then transformed into the global coordinate system using the camera pose matrix. By aggregating the results from all frames, the union of the camera frustums defines the overall scene bounds.
2) **Initialization**
   Initialization defines the spatial boundaries and resolution of the volumetric grid, ensuring accurate scene representation. Once the bounds are established, the volumetric grid is constructed with a resolution defined by the voxel size, which dictates the number of voxels along each axis. Each voxel is initialized with a TSDF value of $+1$, representing no surface within the truncation distance, an initial weight of 0, and the color set to black, indicating no contribution from any frame. The initialization step lays the foundation for subsequent integration, aligning the voxel grid with the observed scene and ensuring computational efficiency.

3) **Frame Integration**
Each frame, comprising the depth map, RGB image, and pose, contributes to updating the TSDF and weight grids. This process involves voxel-to-camera projection, truncated signed distance calculation, and updating the TSDF, weights, and colors. For each voxel, the world coordinates are transformed into the camera coordinate system, where they are then projected onto the image plane to identify the corresponding pixel locations. Voxels whose pixel locations fall outside the image boundaries are discarded during the update process.
The signed distance d between the voxel center and the surface is computed using the following formula:

$$d = D_{\text{pixel}} - D_{\text{voxel}}, \tag{1}$$

where D_{pixel} represents the depth value of the projected pixel from the depth map, and D_{voxel} is the distance of the voxel from the camera along the principal axis. The truncation step involves clipping the signed distance to the range of $[-t, t]$, where t is the truncation margin. This is achieved by normalizing the distance obtained by t and clipping it to the range $[-1, 1]$.

$$t_d = \max(-1, \min(1, d/t)). \tag{2}$$

This truncation improves memory efficiency and focuses the reconstruction on the relevant scene information. After truncating the signed distance, the computed truncated distance value and the weight of the frame are used to update the TSDF.

4) **Update Step**
The TSDF value t_d for a voxel is updated using a weighted average:

$$t'_d = \frac{w \cdot t + w_{\text{new}} \cdot t_d}{w + w_{\text{new}}}, \tag{3}$$

where w_{new} is the frame's contribution weight, t and w are the currently stored values for the TSDF and weight, respectively. The weight grid is also updated:

$$w' = w + w_{\text{new}}. \tag{4}$$

The RGB color of the voxel is updated similarly:

$$C' = \frac{w \cdot C + w_{\text{new}} \cdot C_{\text{new}}}{w + w_{\text{new}}}, \tag{5}$$

where C is the current color, and C_{new} is the color of the voxel based on the given RGB image of the new frame.

5) **GPU Optimization** The CUDA code optimization for the TSDF-based 3D reconstruction pipeline significantly improves the computational efficiency and scalability, enabling real-time processing for large datasets. The code is designed to leverage parallel processing capabilities of modern GPUs, where each voxel update, depth map projection, and mesh generation step is performed concurrently across many threads. Shared memory is utilized for frequently accessed data such as depth maps and voxel information, reducing the need for slower global memory accesses. Furthermore, the occupancy of the GPU is optimized by adjusting block sizes and thread counts to ensure that the maximum number of threads is utilized without causing resource contention. These optimizations collectively enable the pipeline to handle real-time pose estimation and mesh reconstruction, providing high performance for 3D model generation and refinement.

B. Mesh Generation

There are two methods we have used for generating a mesh from the TSDF representation of the 3D object: Marching Cubes and Occupancy Dual Contouring.

- **Marching Cubes** algorithm is a popular method for generating meshes from a TSDF. It works by processing the TSDF in a grid of cubes, where each voxel is classified based on the sign of the TSDF value (inside or outside the surface). The algorithm then interpolates the surface along the edges of the cubes where the TSDF value changes sign, using a precomputed lookup table to generate triangles. This process results in a smooth surface representation but can produce overly detailed meshes, especially at high resolutions.
- **Occupancy Dual Contouring** is an alternative method designed to handle sharp features and generate more efficient meshes. Instead of using edge intersections like Marching Cubes, it places vertices at the centroids of voxel faces that intersect the surface. This allows for better preservation of sharp features and reduces the number of vertices, resulting in a more efficient mesh. Dual Contouring is especially useful for 3D models requiring high fidelity, such as those with sharp geometric details.

C. 6DOF Pose Estimation

Once the TSDF-based 3D model is generated, the next step is to estimate the 6 Degrees of Freedom (6DOF) pose of the object in each new frame. This is done by finding the nearest pose from the frames that have already been used for constructing the TSDF. The nearest pose corresponds to the frame that has the minimum Mean Squared Error (MSE) loss between the TSDF points. Specifically, we compare only the surface points of the TSDF, ignoring the texture, as the goal is to match the shape and spatial positioning of the object. The frame with the smallest MSE loss in surface alignment is considered the closest pose, which provides an initial estimate of the object's 6DOF pose.

Once the initial pose is estimated using this nearest neighbor approach, the generated 3D mesh (obtained from the TSDF) can be fed into the continuation of the pipeline for further refinement. The key next step involves using epipolar geometry to rectify the initial pose. Epipolar geometry helps to align the new frame with the reference frame by considering the geometry of the camera setup. This process adjusts for any misalignment between the initial mesh and the observed scene,

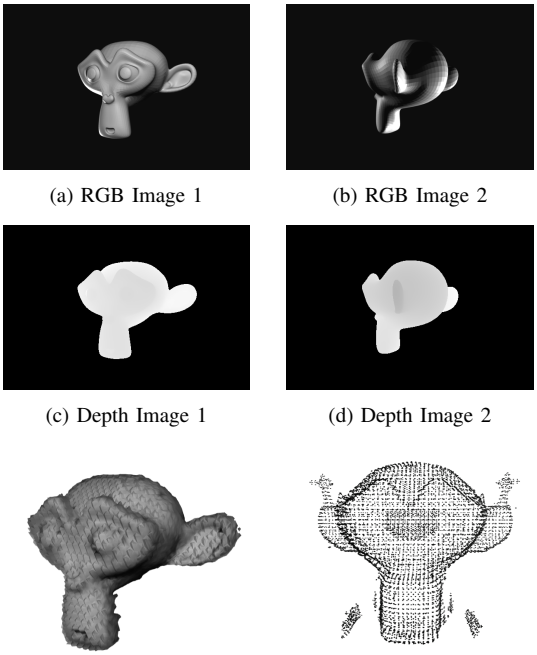
refining the pose to more accurately represent the object's position in the 3D space. Epipolar rectification ensures that the poses are consistent across different viewpoints by adjusting for any camera translation or rotation errors.

After the rectification step, the pose undergoes differential rendering, a technique that compares the silhouette of the rendered 3D model (from the current pose) with the silhouette extracted from the 2D image. This comparison helps to refine the pose by calculating the difference between the predicted and observed silhouette boundaries. The resulting error (or loss) is then backpropagated through the network to update the pose. The gradients from this process help to adjust the pose parameters in order to minimize the discrepancy between the rendered and observed silhouette, ultimately yielding the final, refined 6DOF pose.

This pipeline—starting with nearest pose estimation using TSDF, followed by epipolar geometry rectification and differential rendering—ensures that the 6DOF pose is progressively refined for accurate 3D reconstruction and pose tracking.

V. Results

A. 3D Reconstruction



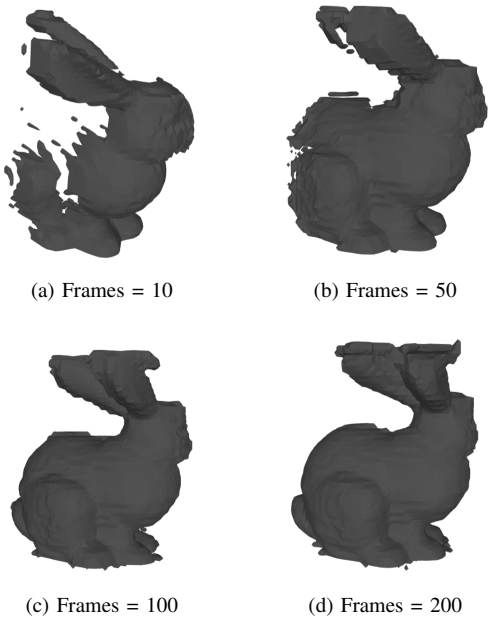
(a) RGB Image 1 (b) RGB Image 2

(c) Depth Image 1 (d) Depth Image 2

(e) Reconstructed Mesh (f) Reconstructed Point Cloud

Fig. 6: 3D Reconstructions using the RGB-D images and camera poses

The figure 6 demonstrates 3D reconstruction using RGB-D images and camera poses. RGB images (a, b) and corresponding depth images (c, d) are used to generate a reconstructed mesh (e) and point cloud (f). The mesh represents the object surface, while the point cloud provides a spatially discrete representation for visualization and analysis.



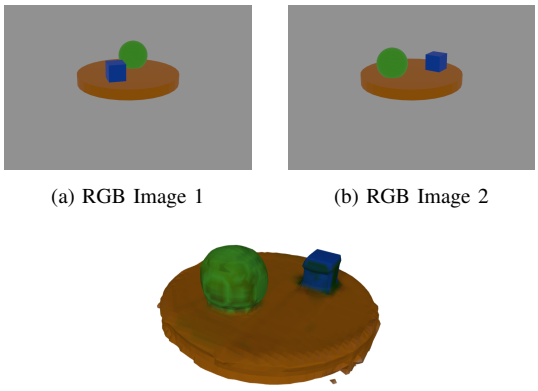
(a) Frames = 10 (b) Frames = 50

(c) Frames = 100 (d) Frames = 200

Fig. 7: Update TSDF with more frames.

The figure 7 shows the meshes of the Bunny dataset as it gets constructed, with the number of frames increasing from 10 to 200. The impact of incorporating more data on the object's representation helps in adding more information resulting in better reconstructions..

B. Textured Mesh Reconstruction



(a) RGB Image 1 (b) RGB Image 2

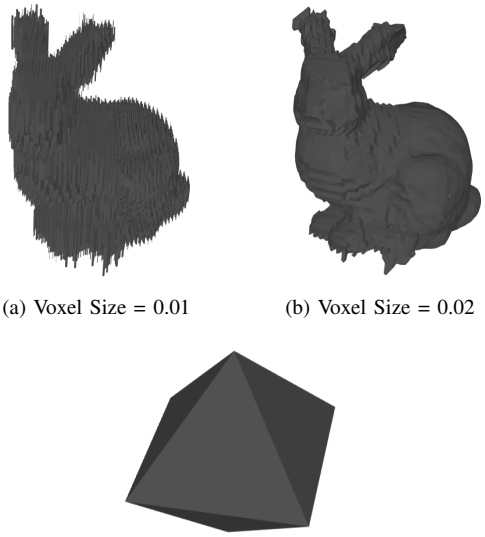
(c) Mesh for Textured Scene

Fig. 8: Reconstructed mesh of the Table with objects dataset

The concept of storing and updating TSDF values can be expanded to include updating the color value at each voxel. A Marching Cubes-like algorithm can then be used to determine the color for each mesh vertex through interpolation techniques. Figure 8 presents the colored reconstructed mesh derived from the RGB images.

C. Comparison of TSDF Hyper-parameters

The hyper-parameters in TSDF construction are the voxel size and the truncation distance.

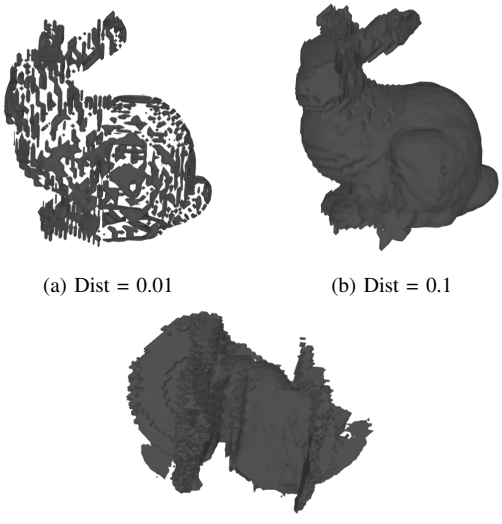


(a) Voxel Size = 0.01 (b) Voxel Size = 0.02

(c) Voxel Size = 0.5

Fig. 9: Difference in meshes on varying Voxel Sizes

Figure 9 shows the quality of reconstructed meshes with varying voxel sizes. Smaller voxel sizes result in finer details which might also appear as nice but increase computational complexity, while larger voxel sizes reduce detail, leading to a coarser mesh. This highlights the trade-off between reconstruction accuracy and efficiency when selecting voxel resolutions for 3D reconstruction tasks.



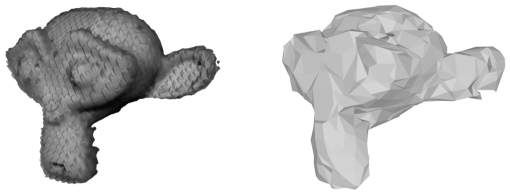
(a) Dist = 0.01 (b) Dist = 0.1

(c) Dist = 0.5

Fig. 10: Difference in meshes on varying Truncation Distances

Figure 10 illustrates the impact of varying truncation distances on TSDF-based reconstruction. Smaller truncation distances preserve finer surface details but may result in incomplete reconstructions near sharp edges. In contrast, larger truncation distances smooth out surfaces but risk losing detail. Balancing truncation distance is crucial for accurate and efficient reconstruction.

D. TSDF to Mesh Methods



(a) Marching Cubes (b) Occupancy Dual Contouring

Fig. 11: Comparison of TSDF to Mesh Methods

Figure 12 shows that Marching Cubes generates a denser and noisier surface representation, while Occupancy Dual Contouring produces a cleaner, lower-polygon mesh. However, Occupancy Dual Contouring (OCD) is computationally expensive, requiring significantly more processing time compared to Marching Cubes. This trade-off makes Marching Cubes more suitable for applications demanding quick reconstruction, while OCD is better for scenarios requiring approximate and simplified surfaces.

E. Scene Reconstruction

The process of reconstructing 3D objects from general images can also be applied to scenes where different views capture various parts of the environment. All the information from various areas of a scene can be integrated into the TSDF. Figure **??** illustrates the reconstructed mesh of a kitchen from the 7-Scenes Dataset.

F. Performance Analysis

	Stanford Bunny	**Suzanne**	**Table with object**
Frame Count	500	500	500
Time Taken (s)	1.68	0.45	1.68
FPS	296.67	530.74	296.67

TABLE I: Speed comparison for different objects.

The results presented in the Table I highlight a significant improvement in performance when comparing the execution times for three different objects: Stanford Bunny, Suzanne, and Table with Object. The CPU implementation achieved a mere 0.5 FPS, which highlights a stark contrast in processing power. The speedup achieved by the GPU implementation is substantial—nearly 600 to 1000 times faster than the CPU version, showcasing the enormous advantages of using GPUs for computationally intensive tasks like rendering and object processing.

(a) Reconstructed Kitchen Mesh

(b) Original Kitchen Image from Dataset

Fig. 12: Comparison of TSDF to Mesh Methods

VI. Conclusion

This work presents a robust and efficient pipeline for real-time 3D reconstruction and 6-DoF pose estimation using Truncated Signed Distance Functions (TSDF). The proposed methodology combines advanced techniques, including volumetric representation, mesh generation, and pose refinement, to achieve high accuracy and scalability. Through extensive evaluation on real-world and synthetic datasets, the approach demonstrates significant advantages in handling dynamic scenarios, such as occlusions and texture variations, while achieving state-of-the-art performance metrics.

The integration of GPU optimization enables real-time processing, significantly improving computational efficiency compared to CPU-based implementations. Moreover, the comparative analysis of mesh generation techniques and TSDF hyperparameters highlights the trade-offs between quality and computational demands, providing valuable insights for various application contexts.

Overall, this pipeline establishes a foundation for future advancements in applications such as augmented reality, robotics, and autonomous systems, emphasizing its potential for real-time deployment and further refinement.

VII. Contributions

Ansh Chablani: Dataset Generation, Pose Estimation

Gaurav Behera: TSDF construction, GPU optimization

Samkit Jain: TSDF to mesh (Marching Cubes, ODC), visualization

References

[1] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[2] Xingyi He, Jiaming Sun, Yuang Wang, Di Huang, Hujun Bao, and Xiaowei Zhou. Onepose++: Keypoint-free one-shot object pose estimation without CAD models. In *Advances in Neural Information Processing Systems*, 2022.

[3] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021.

[4] Jiaming Sun, Zihao Wang, Siyu Zhang, Xingyi He, Hongcheng Zhao, Guofeng Zhang, and Xiaowei Zhou. OnePose: One-shot object pose estimation without CAD models. *CVPR*, 2022.

[5] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Müller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–617, 2023.

[6] Diana Werner, Ayoub Al-Hamadi, and Philipp Werner. Truncated signed distance function: experiments on voxel size. In *Image Analysis and Recognition: 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part II 11*, pages 357–364. Springer, 2014.