NERF BASED EFFICIENT CONTINUOUS 3D RECONSTRUCTION USING SLAM

Thesis submitted to
Visvesvaraya National Institute of Technology, Nagpur
in fulfillment of requirement for the award of
degree of

Bachelor of Technology
In
Electronics and Communication Engineering

by

Bipasha Parui (BT19ECE019)
Yagnesh Devada (BT19ECE122)

under the guidance of

Dr. K. Surender

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)

2022-2023

NERF BASED EFFICIENT CONTINUOUS 3D RECONSTRUCTION SING SLAM

Thesis submitted to
Visvesvaraya National Institute of Technology, Nagpur
in fulfillment of requirement for the award of
degree of

Bachelor of Technology
In
Electronics and Communication Engineering

by

Bipasha Parui (BT19ECE019)
Yagnesh Devada (BT19ECE122)

under the guidance of

Dr. K. Surender

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology
Nagpur 440010 (India)

2022-2023

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology, Nagpur

DECLARATION

We, "Bipasha Parui" and "Yagnesh Devada" hereby declare that the project work titled "**NeRF Based Efficient Continuous 3D Reconstruction sing SLAM**" is carried out by us in the Department of Electronics and Communication Engineering of VNIT, Nagpur, India. The work is original and has not been submitted earlier as a whole or in part for the award of any degree/diploma at this or any other Institution/University.

Bipasha Parui BT19ECE019

Yagnesh Devada BT19ECE122

Date: 04-05-2023

CERTIFICATE

This is to certify that the thesis titled "**NeRF Based Efficient Continuous 3D Reconstruction sing SLAM**" submitted by **Bipasha Parui** and **Yagnesh Devada** in partial fulfilment of requirements for the award of Bachelor of Technology in the Department of Electronics and Communication Engineering of VNIT, Nagpur. The work is comprehensive, complete and fit for evaluation.

Dr. K. Surender
Assistant Professor,
Electronics and Communication Engineering,
VNIT,Nagpur

Dr. A.G.Keskar
Professor and Head,
Electronics and Communication Engineering, VNIT,Nagpur
Date: 04-05-2023

Department of Electronics and Communication Engineering
Visvesvaraya National Institute of Technology, Nagpur

DECLARATION

We, "Bipasha Parui" and "Yagnesh Devada" hereby declare that the project work titled "**NeRF Based Efficient Continuous 3D Reconstruction sing SLAM**" is carried out by us in the Department of Electronics and Communication Engineering of VNIT, Nagpur, India. The work is original and has not been submitted earlier as a whole or in part for the award of any degree/diploma at this or any other Institution/University.

Bipasha Parui BT19ECE019

Yagnesh Devada BT19ECE122

Date: 04-05-2023

CERTIFICATE

This is to certify that the thesis titled "**NeRF Based Efficient Continuous 3D Reconstruction sing SLAM**" submitted by **Bipasha Parui** and **Yagnesh Devada** in partial fulfilment of requirements for the award of Bachelor of Technology in the Department of Electronics and Communication Engineering of VNIT, Nagpur. The work is comprehensive, complete and fit for evaluation.

Dr. K. Surender
Assistant Professor,
Electronics and Communication Engineering,
VNIT,Nagpur

Dr. A.G.Keskar
Professor and Head,
Electronics and Communication Engineering, VNIT,Nagpur
Date: 04-05-2023

Abstract

Ever since autonomous vehicles began to emerge, the need for reliable and accurate mapping and localization techniques has become increasingly important. This is achieved via Simultaneous Localization and Mapping aka SLAM which is a technique used in robotics and computer vision to allow a robot to create a map of its surroundings while simultaneously determining its location within that environment. 3D reconstruction or 3D mapping of an environment is one of the most crucial stages of Simultaneous Localisation and Mapping (SLAM). Various types of sensors such as Inertial Measurement Unit(IMU), LiDAR and Cameras are used independently or by fusion that can be used for mapping the environment space. Of these, Camerasimages or RGB images provide not only geometric information about the environment but also semantic information about the objects in the scene. This semantic information allows for more detailed and accurate 3D mapping, as the robot can differentiate between different objects and surfaces, and use this information to construct a more detailed map. Additionally, cameras are relatively inexpensive and widely available, making them a practical choice for many applications. Different mapping methods can be used to achieve a 3D environment space such as voxel-based, mesh-based, volumetric or point based. Of which point based and volumetric methods are the most suitable for applications involving localization and navigation aided by 3D maps. This thesis project focuses on 3D reconstruction or 3D mapping of the environment using Neural-Radiance fields (NeRF) to overcome issues such as point sparsity observed in point clouds. We provide more consistent 3D maps that can be used to enhance SLAM in its localization stage.

List of Figures

List of Tables

List Of Publications

- Bipasha Parui, Yagnesh Devada, K. Surender. "Vision based 3D mapping - From traditional to NeRF based approaches". In: *2023 2nd International Conference on the Paradigm Shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS), Nagpur, India, 2023*

Contents

Chapter 1

Introduction

Ever since the idea of autonomous navigation was born, Simultaneous Localisation and Mapping (SLAM) [2] has been used to estimate sensor motion, navigating through and reconstructing 3D maps of unknown environments. SLAM can be defined as having two major parts; building a map of an unknown indoor or outdoor environment and tracking the position or movement of the sensors and camera (generally on a mobile robot) at different time steps.

During the 3D mapping process, data from various sensors such as cameras, LiDAR, and inertial sensors are used to create a three-dimensional model of the environment.
The 3D mapping process in SLAM typically involves four main steps: data acquisition, data processing, map construction, and map optimization.

- In the data acquisition stage, the data is collected, in our case RGB images from a camera and the robot moves in its environment. The data is captured in a series of time stamps.

- In the data processing stage, the raw sensor data is pre-processed and filtered to remove noise and other unwanted artefacts. Other types of maps such as image normals, image depths etc may also be calculated as required by different 3D reconstruction algorithms.

- In the map construction stage, the processed data is used to create a 3D representation of the environment. This may involve using methods such as voxel-based mapping, point cloud-based mapping, or NeRF-based mapping to create a detailed and accurate map.

- The map optimization stage, the map is refined and improved based on feedback from the robot's sensors. This may involve using techniques such as loop closure

detection, which involves identifying and correcting errors in the map caused by the robot's movement.

Mapping of an environment is mostly seen as the main outcome of a SLAM system but it also plays a parallel role in tracking and localizing a robot in the global map. The mapping stage is important for several reasons.

- It can be used for planning and navigation. For example, an autonomous vehicle can use the map to plan a route and avoid obstacles in its path.
- Secondly, mapping allows the robot to understand its surroundings and detect changes in the environment. This is important for tasks such as object recognition and tracking which are essential for obstacle avoidance in dynamic environments.
- Finally, the map can be used for localization, which is necessary for accurate robot navigation and control.

There are various methods that can be used for 3D mapping in SLAM, including voxel-based mapping, point-based mapping, and NeRF-based mapping. While the most widely used method is Point-based mapping which involves creating a point cloud from the sensor data they usually suffer from two main challenges. Firstly, it suffers from outliers and redundancies due to repeated point allocation with minute coordinate changes for the same scene. Secondly, most SLAM and 3D reconstruction systems have free spaces in 3D maps due to partial occlusion, non-textured surfaces, illumination changes, etc.

NeRF-based [6] mapping is a relatively new approach that combines the advantages of both voxel-based and point-based mapping. It involves training a deep neural network to predict the radiance at any point in 3D space, given a set of 2D images and their corresponding camera parameters. This approach allows for the creation of highly detailed 3D maps that capture both geometric and photometric information about the environment. NeRF-based maps can provide a high level of accuracy and detail, but maybe computationally intensive and require large amounts of data.

An accurate 3D map aids in optimizing the camera pose of the current frame via point reprojection. This optimized camera pose in turn helps in adding new scene points to the map as seen by that camera frame which is then used to globally update the overall 3D map.

Most of the deep learning-based vSLAM methods do not focus on solving outliers and gaps or exploring reconstructions beyond single objects. This trend was broken

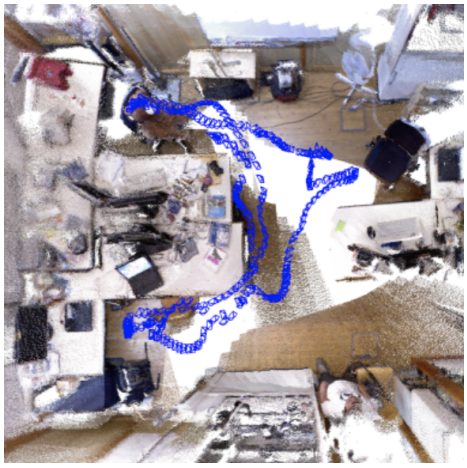

Figure 1.1: 3D mapping and localization

with the advent of Neural Radiance Fields, an MLP-based novel view synthesis which explores 3D scene representations as neural implicit fields. Such a representation can be easily visualised and navigated in point cloud formats unlike previous mesh or voxel-based methods. NeRFs-based SLAM systems and other mapping methods show remarkable consistency in maps with the size of a few tens of megabytes.

In this project we construct a 3D map of the environment using Neural Radiance Field. The ability of NeRF-based 3D maps to generate novel views can be used to improve SLAM tasks and robot relocalization problems in several ways. The system generates novel views of the environment from the robot's current position and uses these to match sensor data to the 3D map more accurately. The use of neural networks allows better prediction capabilities by observing features in images that can help to improve the accuracy of 3D maps by filling in missing information or correcting errors that may be caused due to illumination issues or occlusions.

Chapter 2

Literature Review

The summary of the research papers studied to understand the state-of-the-art concepts has been presented here:

2.1 Feature based methods

Feature-based visual mapping algorithms are one of the earliest ones to be developed. They are divided into two broad categories Filter-based and Feature-based. Filter-based methods like MonoSLAM[1] and its variants [7] make use of Kalman Filters.

Feature-based algorithms divide the whole process into prediction and correction steps. The prediction step guesstimates the position of the camera while the correction step aims to correct it. While the feature-based method uses feature points extracted from frames using SIFT, SURF, ORB, etc, which are matched with consecutive frames to find their transformation and thus find the overall trajectory of the camera and the map of an unknown environment. At the next stage, the localization module makes use of epipolar constraints (coplanarity constraints) to find the relative transformation between two consecutive frames that could be triangulated to find their corresponding 3D points in a local region. This process is iterated over multiple frames until a sufficient number of points get accumulated to form a local map which is then merged with the global map. Various optimization methods like Pose Graph Optimization (g2o) [4] and Bundle Adjustment are used to optimize both the pose and the map to minimize the reprojection error.

2.1.1 Existing works in Feature-Based/Indirect methods

MonoSLAM [1] was one of the first monocular SLAM frameworks to be developed. It employs an extended Kalman Filter for localisation and building the map of an unknown environment. The prediction step uses a linear motion model employing the basics of Newtonian mechanics assuming constant linear and constant angular velocities. However, since it tracks all previously visited points despite their absence in the local map, MonoSLAM becomes computationally expensive which increases exponentially with the size of the map. Due to this reason, the number of feature points that are detected in a frame is kept to be limited, leading to sparse points

per local map. However, since these 3D points in the local map are used to make data associated with a global map to remove common points, the sparse nature of the map leads to high map updation time.

PTAM [10] One of the major turning points in the development of SLAM was PTAM which proposed the use of two different threads for the Tracking and Mapping algorithm. This ensures that the mapping thread which is more computationally expensive does not interfere with the tracking module while also making the whole process faster. The mapping stage of PTAM is divided into frames and keyframes. Where a frame is a single image from the camera feed containing features, a keyframe is a "milestone", a representative frame for multiple neighbouring frames in terms of features and camera position. This approach is important because, in the case of tracking loss or a significantly large deviation in the delta transformation between frames, it can revert to the state of the last keyframe and revive the SLAM system. PTAM makes use of both local and global bundle adjustments in order to improve the accuracy of the map by optimizing the generated local map and pose of keyframes respectively. As a result, PTAM leads to denser maps as compared to MonoSLAM.

ORB-SLAM [8] was another milestone in the development of SLAM algorithms. It is a real-time monocular SLAM algorithm which can run efficiently on a CPU in a large environment. It is a significant improvement over limiting factors of PTAM such as lack of loop closure, low viewpoint invariance and human intervention for map bootstrapping. It uses Pose Graph Optimizations (g2o) to minimize reprojection errors for optimizing keyframe and 3D point poses. ORB-SLAM successfully implements loop closure via a spanning tree for detected loops and a co-visibility graph for strong edges maintained by the system.

While map initialization is done using triangulation further feature matching is realised using a 3-step process.

- Projecting the map points (taking into consideration the field of view of the camera) onto the current frame and finding the nearest neighbour.
- Projecting the triangulated feature points present in the previous frame onto the current frame and finding the nearest neighbour.
- Epipolar search.

This allows a robust dense map to be created, unlike the previous two methods.

2.2 Direct based method

Direct methods are different from its counterpart in the sense that they do not use features in the image but instead use every pixel to find the inter-frame transformation and generate the map point in the local map. The camera's motion is tracked using image alignment algorithms. Since this method uses every pixel for localisation and mapping it tends to generate a dense and accurate map of the environment. Due

to the higher pixel density being used, it has a higher occlusion handling capacity than feature-based methods.

Dense Tracking and Mapping(DTAM) [9] uses the general structure laid out by Parallel Tracking and Mapping (PTAM) but an "upgraded" version for the direct method. The tracking module works by finding the image matching by using iterative methods like Gauss-Newton to find the relative transformation between two frames. Additionally, it also uses the information from the map to improve its accuracy. DTAM assumes that the corresponding pixels in consecutive frames have the same if not nearly the same intensity. To estimate the depth of each pixel they use a global energy minimisation framework. Here the energy is defined as the summation of photometric errors over all frames.

Large Scale Direct SLAM [3] is a semi-direct monocular SLAM method for large-scale environments which can run in real-time on a CPU. It proposes a new scale-aware method for merging local and global maps taking care of the depths. This method incorporates another module for map optimization using a 7 DoF pose-graph base optimization method employed to obtain a geometrically consistent map, which was previously missing in DTAM. The tracking module sub-samples the pixel with high gradients allowing it to ignore all the textureless regions and only focus on regions with more visual information. While finding the depths, initially random values are set as an initial depth value for each pixel and they are then found by iteratively using a Gauss-Newton method by minimizing the photometric error. A major drawback of this method is that it does not focus on regions with a low gradient which includes textureless walls and other flat surfaces which define the geometry of the environment.

BADSLAM [12] is a method that uses surfels and keyframes to represent the map, reducing the amount of data for optimization (local bundle adjustment) however, still finds features using the direct method. Although a surfel-based method the paper is fundamental as it proves that direct RGB-D SLAM systems are highly sensitive to rolling shutter, RGB and depth sensor synchronization, and calibration error. BAD-SLAM is the first to establish a dense BA approach for SLAM using an RGB-D camera.

2.3 DL based methods

Deep learning in recent years has been used extensively to solve SLAM systems by making separate sub-modules of SLAM differential, such as feature detection, pose estimation, loop detection, depth estimation, semantic aided mapping etc. Works such as CNN-SLAM, Code-SLAM and DVSO introduce depth estimation based on CNN in the framework of LSD-SLAM to be used for improving traditional pose estimation and mapping accuracy. CNN-based work of Zhu et al and others focus mainly on loop detection by use of Bag of Words, semantic sub-graphs etc. D3VO

uses DL for three aspects, depth estimation, pose estimation, and uncertainty estimation. LSTM and GRU variants of RNN have been used either independently as in Droid-SLAM or in conjugation with CNN DeepSeqSLAM to create a series of depths and enable end-to-end pose and sequence positional learning for pose estimation respectively. While most of these works provide better depths and features that ensure good mapping in later stages, they do not dedicatedly focus on improving 3D maps.

An interesting combination of 2D and 3D semantic information with 3D point clouds has been seen to aid better navigation in reconstructed maps of SLAM. Most of the traditional SLAM mappings are based on geometric matching at the pixel level. Semantic information upgrades data association from pixel level to object level, thereby improving the accuracy of complex scenes. Zhao et al. and many further works introduce a landmark visual semantic SLAM system for a large-scale outdoor environment. At its core, it combines 3D point clouds in ORB-SLAM with semantic segmentation information in YOLO3 based CNNs to associate semantic point clouds with architectural landmarks. While these works give point cloud map better perceptual ability under high camera rotation and strong illumination, adding an additional module to existing SLAM drastically slows down the system. Moreover, they do not directly optimize point clouds but rather provide additional feature masks to counteract challenges in traditional point clouds.

2.4 NeRF Methods

Neural Radiance Field (NeRF) [6] create 3D implicit representations of a scene for novel view synthesis. NeRFs are very low cost in storage and have easy reproducibility of maps. Additionally, NeRF models are self-supervised and unlike many other 3D neural representations NeRFs do not require 3D/depth supervision. It may not only be used for mapping but also to optimize point clouds or estimate poses using Structure from Motion (SfM). NeRF's ability to generate a 2D novel view of the 3D environment given an unobserved pose can be used to solve robot re-localization problems.

As the NeRF model functions differently from classical or previous deep learning methods, we have first explained the standard NeRF formulation in the section below before going in-depth about the current works that explore 3D mapping and reconstruction.

2.4.1 Works

PointNeRF [15], shows the capability of neural implicit representations in removing outliers and holes in occluded, unobserved regions as seen in most point clouds. While traditional NeRF papers construct radiance fields using global MLPs, PointNeRF uses 3D neural points with each neural point encoding the local 3D scene geometry and appearance in its neighbourhood, to model a scene via differential ray marching. Using MVSnet (3D deep multi-view stereo) as the initial module to generate an initial point cloud, PointNeRF further uses kNN networks to grow neural points in empty regions. Although the method is fundamental, it is limited to closed-loop indoor or

small scenes which are not sufficient as most vision robots navigate in open, long, outdoor environments. This development was seen in URF which explores outdoor environments using sparse LiDAR scans instead of heavy SFM-modeled point clouds.

iMAP [13] One of the earliest works that explore the NeRF-SLAM system is iMAP, a hybrid of classical and deep learning stages to achieve a dense real-time SLAM system. It uses an implicit neural scene representation via continual learning to jointly optimize camera poses and a full 3D map. It is one of the first works that explore online training of networks without any prior data, building an implicit 3D model of occupancy and colour which is also immediately used for tracking. In order to give attention to information-sensitive areas or where reconstruction is not yet precise, iMAP uses a bounded window of a sparse set of representative keyframes that are incrementally selected based on information gain.

While iMAP does joint mapping and tracking optimization iteratively by maintaining a loss distribution over all registered keyframes, another paper NICE-SLAM takes a different approach by selectively including only those frames which have visual overlap with the current frame when optimizing the scene geometry.

NiceSLAM [18] The key limiting factor of iMAP stems from its use of a single multi-layer perceptron (MLP) to represent the entire scene, which can only be updated globally with every new, potentially partial RGB-D observation, which is tacked by Nice-SLAM by introducing encoder-decoder structures for local grid based neural implicit encoding in a fine to coarse manner. These two changes not only ensure the geometry outside of the current view remains static but also result in a very efficient optimization problem as we only optimize the necessary parameters each time. However, the only downside of NICE-SLAM is that it makes use of voxel-based representations for feature grid formation thereby increasing storage cost.

MeSLAM [5] However, both continue to suffer from map-forgetting issues resulting from the use of MLPs only. This is solved by MeSLAM by making each network's parameters implicitly represent a map of the region (local map). These are further stitched together much similar to traditional SLAM systems for better mapping. This strategy provides local maps stitched into a global one without distorted overlapping and deformation.

'Vision only NeRF navigation' is a minimal yet important paper as it explains dedicated autonomous planning and navigation through a NeRF environment using only RGB vision. It does so by comparing viewed images with NeRF-synthesized novel views for a said pose at each step.

iNeRF [16] Another noteworthy work that has much potential is iNeRF which works in inverting the NeRF formulation. iNeRF can use a NeRF model to estimate 6 DoF pose for scenes and objects with complex geometry. An important feature of iNeRF is the removal of 3D model dependency for pose estimation which often leads to higher costs. The paper also talks about interest-based pixel sampling as opposed

to full image (all pixels) analogous to direct methods but in a more selective manner instead of keypoints/feature-based methods.

BlockNeRF [14] Large-scale projects like BlockNeRF take liberty with computation aiding in the creation of entire cities of neural 3D representations giving competition to the state-of-the-art point cloud city maps with the use of multiple NeRF models for separate sections of city/large environment. Despite large computations, their storage and reconstruction costs are much lower than direct point cloud storage methods.

In reference to mapping, however, NeRFs have shown remarkable capability in reducing not only 3D scene representation storage costs but also the ability to prune and grow sparse, occluded regions. They have also shown the ability to predict points in unobserved and occluded parts of the environment as opposed to previous methods. Furthermore, once trained NeRF-based maps are very low-cost when it comes to storage, map reuse and navigation.

Chapter 3

Methodology

3.1 SLAM

3.1.1 Motivation

Simultaneous Localization and Mapping (SLAM) is a robotics and computer vision technique that allows a robot or vehicle to move and interact with its environment in real-time without using external sensors or GPS. SLAM's goal is to map the surroundings while also estimating the location and orientation of the robot inside that map.

The SLAM algorithm uses a variety of sensors, such as cameras, LiDARs, and IMUs, to gather information about the surroundings and the robot's movement. The position and orientation of the robot are then estimated using this sensor data in real-time, along with the location and properties of objects in the immediate surroundings. As the robot moves and new data is gathered, the algorithm can use this estimated information to continuously update and enhance the map of the environment.

SLAM can be divided into two main groups: direct SLAM and feature-based SLAM. In feature-based SLAM, the algorithm recognises and keeps track of distinctive environmental features, such as edges or corners, and makes use of this data to ascertain the position and orientation of the robot. The 3D structure and appearance of the environment are instead inferred directly from the raw sensor data in direct SLAM, which does not need the detection of specific features.

Simultaneous Localization and Mapping (SLAM) is a technology that makes it possible for autonomous systems, including robots, drones, and self-driving cars, to navigate and interact with their surroundings in real time without the aid of external sensors or GPS.

For these systems, SLAM is a crucial technique since it enables them to map their surroundings while also figuring out where they are in relation to that map. As a result, autonomous systems are better equipped to plan their actions and motions and navigate through complex, dynamic settings with increased safety and accuracy.

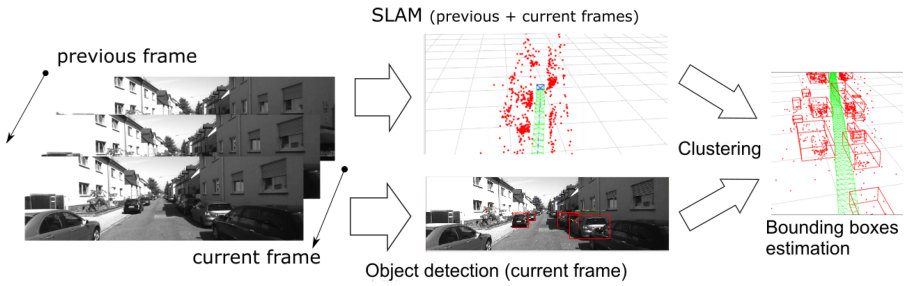

Figure 3.1: The framework employs object detection and visual SLAM to simultaneously estimate both the environment's map and the agent's localization. By utilizing the relative transformation between frames, the current position of the agent within the environment can be determined. (Mazurek et al. 21)

3.1.2 Camera Model

The camera model is a mathematical representation of how a camera maps 3D points in the world to 2D points on an image plane. This transformation depends on several camera parameters such as sensor size, aperture, focal length, and lens distortion. One of the most commonly used camera models is the pinhole camera model, where all light rays from a scene pass through a single point in space, called the projection centre or camera centre before being projected onto the image plane.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where $[u\, v]^T$ is the normalized projected coordinate, with λ as a scale parameter, of the 3D world coordination $[X\, Y\, Z]^T$ in the image plane. This transformation is defined by the calibration matrix K.

The calibration matrix encodes the information about the camera.

$$K = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y are the focal lengths of the lens along the u and v axis of the image plane respectively. (c_u, c_v) is the projection centre of the camera in the image plane and s is the skew coefficient, non-zero if the image axes are not perpendicular.

3.1.3 Camera Calibration

Camera calibration is the procedure of determining the camera matrix, which consists of intrinsic and extrinsic camera parameters. The extrinsic camera parameters specify the camera's position in the world coordinate system and transform the points from the world coordinate system to the camera coordinate system. Meanwhile, the intrinsic camera parameters, represented by the calibration matrix K, include internal parameters such as focal length, field of view, aperture, etc. These parameters

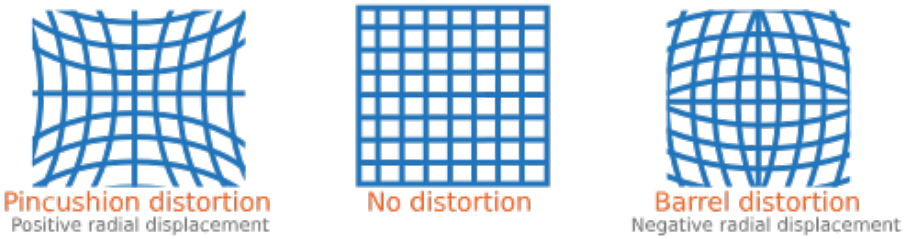

Figure 3.2: Types of distortion: Positive radial displacement, No Distortion and Barrel Distortion

transform the points from the camera coordinate system to the image plane and ultimately to the pixels. One of the widely used camera calibration methods is Zhang's Method [17].

Its steps include:

1. Using the camera, capture many photos of a calibration pattern from various points and orientations. A checkerboard, a grid of dots or circles, or any other pattern with established dimensions might be used as the calibration pattern.

2. Using a corner detection method, such as the Harris corner detector, find the calibration pattern's corners in each picture.

3. Create a list of 3D points for each picture that match to the calibration pattern's observed corners. These points' 3D coordinates should be known in a common coordinate system.

4. Use Zhang's approach to calculate the camera's intrinsic and extrinsic parameters. Zhang's technique estimates the intrinsic parameters using a set of linear equations and the extrinsic parameters using a nonlinear optimisation procedure.

5. Zhang's approach requires at least two photographs of the calibration pattern to determine the intrinsic parameters. The identified corners are utilised to generate a set of equations in terms of the camera settings for each picture.

6. Zhang's approach needs a minimum of three photographs of the calibration pattern in order to estimate the extrinsic parameters. The nonlinear optimisation approach is used to determine the camera location and orientation that best aligns the 3D points with their corresponding image points. The 3D coordinates of the identified corners are translated into the camera coordinates.

7. After estimating the intrinsic and extrinsic parameters, the calibration's correctness may be assessed by projecting the 3D points onto the picture plane using the calculated parameters and comparing them to the detected image points.

8. The calculated parameters can be employed in a wide range of computer vision applications, including 3D reconstruction, object tracking, and augmented reality.

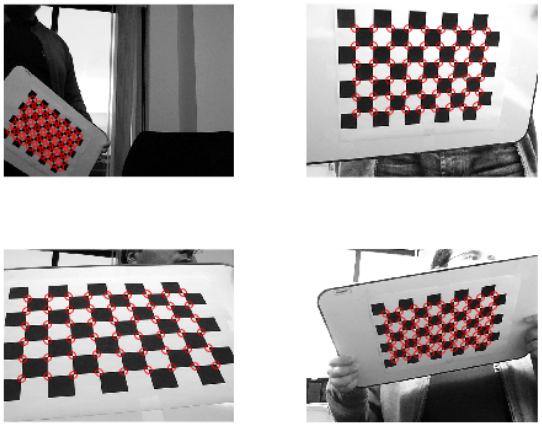

Figure 3.3: Checker Board pattern captured from a stationary camera at different checkerboard angles.

3.2 Localization

Simultaneous Localization and Mapping (SLAM) localization is the process of determining a robot or vehicle's position and orientation inside an environment that has already been mapped out.

In order to precisely detect its position and orientation within the environment, which is essential for preparing its motions and actions, the robot must be able to localise. To do this, the SLAM system typically combines sensor data from cameras and lidars with odometry data, which follows the movements of the robot, to determine the robot's current position.

Localization is a crucial element in SLAM that functions in tandem with mapping. The localization offers the most recent estimate of the robot's position in the map, whereas the map serves as the reference frame against which the robot's present position is calculated. As the robot moves through the environment and collects more sensor data, mapping and localization are iteratively improved, resulting in increasingly precise estimates of the robot's position and the map of the environment.

In SLAM (Simultaneous Localization and Mapping), localization can be accomplished using a number of approaches, such as:

1. **Probabilistic filter-based** methods are a popular state estimation and tracking method that can also be utilised for localization in SLAM. To estimate the robot's current state, it blends noisy sensor measurements with past knowledge of the system dynamics. Filters like Kalman Filter, Extended Kalman Filter, Particle Filter and etc can be used for the same.

2. **Graph-based approaches** is a method in which the environment and robot postures are represented as nodes in a graph, and the robot's measurements and motions are represented as edges between these nodes. The robot's posture may be calculated by optimising the graph based on the measurements and motions.

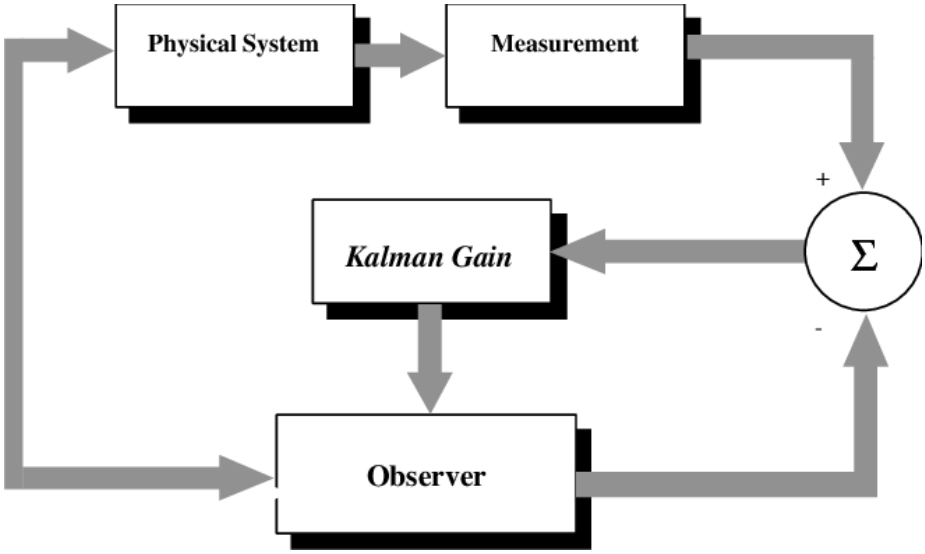

Figure 3.4: Basic Block Diagram of Kalman Filter and its varients.

3. **Visual Odometry** is a technique that calculates the motion of a robot by analysing successive pictures acquired by a camera. It evaluates the relative motion between two successive frames and, using these estimations repeatedly updates the robot's location.

4. **Laser Scan Matching** is a LiDAR-based SLAM approach. It entails comparing successive laser scans to estimate the robot's location and velocity. The scans are aligned by determining the transformation that reduces the space between them.

3.2.1 The Kalman Filter

The Kalman filter is a mathematical algorithm that has broad applications in various fields, including finance, control theory, and robotics. It is based on Bayesian probability theory, which combines uncertain measurements with prior knowledge to estimate the state of a system accurately. The filter is commonly used for state estimation and tracking purposes, where it helps to predict future states based on past observations and reduces the impact of noisy measurements.

The basic steps to Kalman Filter are:

1. **Initialization**: The Kalman filter begins with an initial estimate of the state of the system, which is typically based on prior knowledge or initial measurements. The initial estimate is represented as a mean and a covariance matrix.

2. **Prediction**: The Kalman filter uses the system dynamics and control inputs to predict the state of the system at the next time step. This prediction is represented as a mean and a covariance matrix.

3. **Update**: The Kalman filter combines the predicted state with the measurement to update the state estimate. The update step involves calculating the

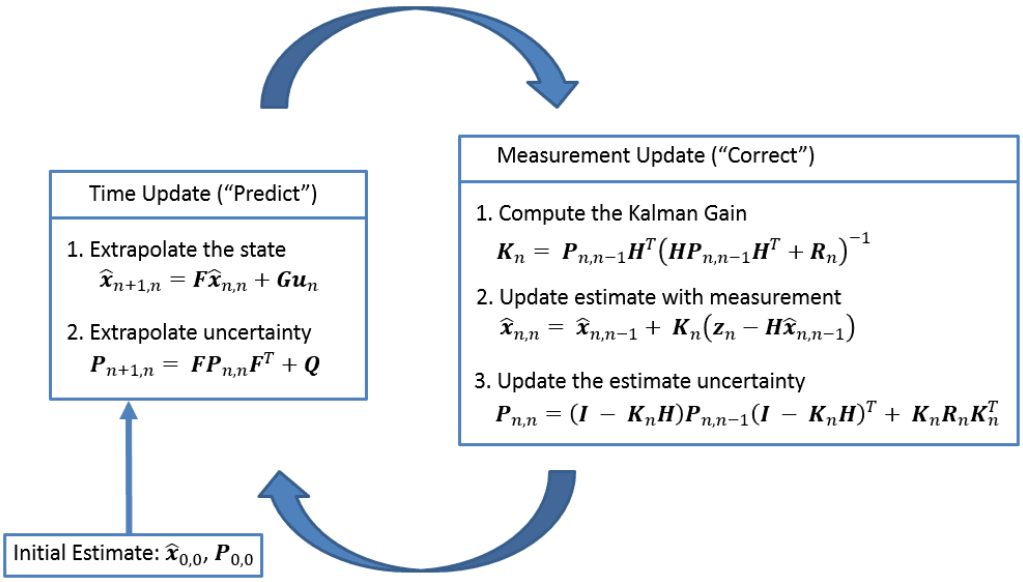

Figure 3.5: Kalman Filter Algorithm

Kalman gain, which is used to determine how much weight to give to the predicted state and the measurement. The updated state estimate is represented as a mean and a covariance matrix.

4. **Repeat**: The Kalman filter continues to repeat the prediction and update steps as new measurements become available. The predicted state becomes the prior for the next time step, and the process is repeated.

While the Kalman filter is widely used and has many advantages, there are also some drawbacks and limitations to be aware of. Some of the main drawbacks of the Kalman filter include:

- The Kalman filter makes the assumptions that the system dynamics and measurement noise are linear and Gaussian, respectively. The filter may not generate reliable results if certain assumptions are not satisfied.
- Computationally demanding: The Kalman filter necessitates substantial processing resources, particularly for large-scale systems. This can make it unsuitable for use in various applications.
- Instability: The Kalman filter is not resistant to outliers or unexpected occurrences. The filter may generate erroneous results if there are major measurement mistakes or disruptions in the system.

3.2.2 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is a Kalman Filter variant that can deal with nonlinear system dynamics and data. EKF approximates the system model and

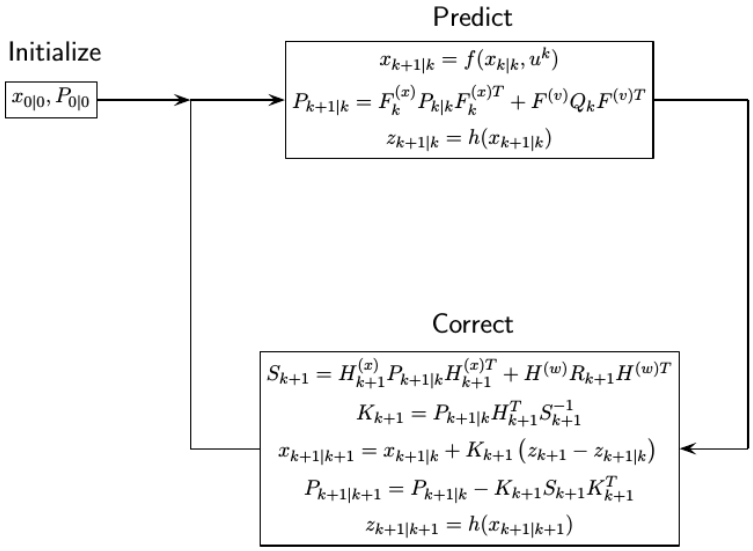

Figure 3.6: The Extended Kalman Filter Algorithm

measurement model as nonlinear functions and linearizes them using a first-order Taylor expansion around the current state estimate.

The EKF may be used in localization to estimate a robot's location and orientation by fusing input from sensors such as wheel encoders, inertial measurement units (IMUs), and range sensors. The EKF maintains a belief in the robot's stance by propagating it forward in time using a motion model and sensor data to adjust for mistakes.

In a mobile robot navigation application, for example, the EKF may be used to estimate the robot's location and orientation by combining data from wheel encoders that measure the distance travelled by each wheel and an IMU that detects the robot's linear and angular accelerations. The EKF estimates the robot's pose using the motion model and then updates the pose estimate using sensor data.

Because it can handle nonlinear system dynamics and measurements, which are common in robotic applications, the EKF is a popular choice for localization. However, the EKF, like the Kalman filter, has limitations and assumptions, such as the requirement for accurate system models and the assumption that measurement noise is Gaussian and additive.

3.2.3 Visual Odometry

Problem Formulation

The problem of recovering odometry estimates from visual readings can be formulated as an agent moves through an area, collecting photographs at discrete time intervals k using a rigidly connected camera system. A stereo camera system consists of two cameras separated by distance known as the *baseline* which is known in advance.

Thus, in case of a stereo system there are a left and a right image at every discrete time instant which can be denoted as $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ and $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$. Figure 3.7 represents the pictorial representation of the visual odometry problem.

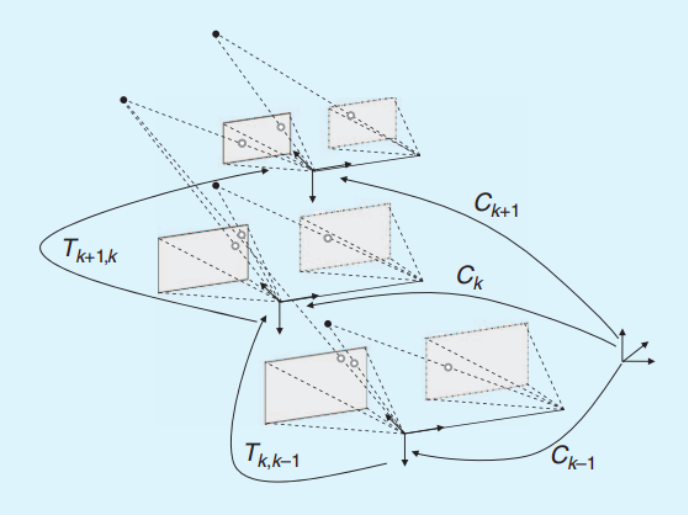

Figure 3.7: Illustration of the visual odometry problem as described in [11].

For simplicity, the camera frame of reference can be assumed to be the same as the agent's frame of reference. In practice, this is often not the case, however, the relative transformation between the agent's frame of reference and camera's frame of reference can be easily computed through external calibration. As indicated in 3.7 two consecutive camera positions can be related by the following rigid body transformation:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

where $R_{k,k-1} \in SO(3)$ and $t \in R^{3x1}$ represent the relative rotation and translation between images captured at time intervals k and $k-1$. Therefore, $T_{k,k-1} \in R^{4x4}$ is called as the transformation matrix and will be represented as T_k for simplicity purposes. Let $C_n = \{C_0, ..., C_n\}$ represent the set of camera poses which respect to the initial coordinate frame at $k = 0$. The goal of the visual odometry problem is to recover this set of camera poses. The pose at the nth time instance, C_n can be recovered by concatenating all transformations upto the nth time instance, that is $T_k(k = 1...n)$. This establishes a recursive update rule with $C_n = C_{n-1}T_n$ where C_0 represents the camera pose at $k = 0$.

The visual odometry pipeline has been summarized in 3.8. For every new pair of images I_k, first features are detected independently in each image. Next, these features are tracked in the next set of images using a local search technique like optical flow. This increases the accuracy of feature matching and reduces the possibility of false matches. The next step consists of computing the relative transformation between discrete time instances k and $k-1$ or the transformation matrix T_k using the spatial location of these feature matches in the respective images. These motion estimation techniques are typically of 3 types: 2D to 2D, 3D to 3D and 3D to 2D. The

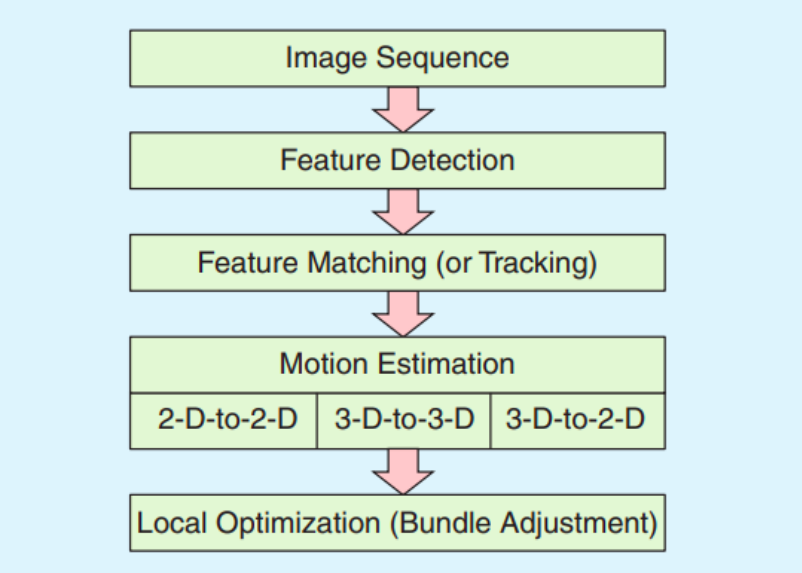

Figure 3.8: Block diagram showcasing the major components of the visual odometry pipeline [11].

camera poses C_k can then be computed by using multiplying the previous camera pose, $C_k - 1$ with the current transformation matrix T_k. We may also opt for iterative refinement methods like bundle adjustment over the last i frames for obtaining a more accurate estimate of the camera trajectory.

3.3 Mapping

Mapping refers to the process of establishing a representation of the environment in which a robot is investigating or navigating in the context of SLAM (Simultaneous Localization and Mapping). Mapping's purpose is to produce an accurate and detailed model of the surroundings that the robot may utilise for navigation and other activities.

Mapping is performed concurrently with localization in SLAM, which entails calculating the robot's location and orientation in the surroundings. The robot collects data about the surroundings using sensors such as cameras, LiDARs, and range finders, and this data is utilised to generate the map.

There are numerous kinds of maps that may be constructed in SLAM to depict the world that a robot is exploring or navigating. The map type chosen is determined by the sensors utilised, the needed precision, and the robot's application. The following are the most common types of maps used in SLAM:

- A **grid map of occupancy** splits the environment into cells and indicates whether each cell is occupied or vacant space. This form of a map is frequently used in conjunction with range sensors like lidars and sonars, which offer measurements of environmental impediments. Occupancy grid maps are extensively utilised in mobile robot navigation, obstacle avoidance, and localization applications.

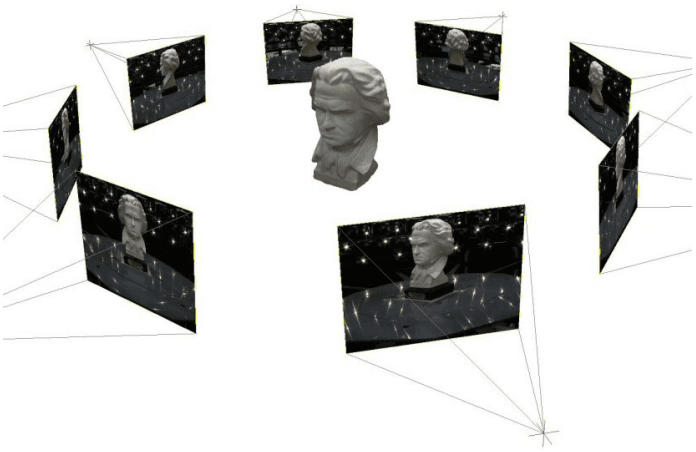

Figure 3.9: 3D Reconstruction

- A **feature map** depicts the environment as a collection of distinguishing elements such as corners or edges. This sort of map is frequently used in conjunction with visual sensors like cameras, which offer pictures of the surroundings. Feature maps are extensively employed in visual navigation, visual tracking, and 3D reconstruction applications.
- A **point cloud** map shows the world as a collection of 3D points, each of which corresponds to a measurement obtained by a range sensor. This form of map is commonly used in conjunction with 3D sensors such as lidars and depth cameras, which offer detailed 3D measurements of the environment. Point cloud maps are extensively utilised in object detection, scene interpretation, and 3D reconstruction applications.

3.3.1 3D Reconstruction

The process of producing a three-dimensional model or representation of an item or environment from two-dimensional photographs or data is referred to as 3D reconstruction in mapping. This may be accomplished by the application of mathematical methods and techniques such as computer vision, photogrammetry, and triangulation. In general, the 3D reconstruction process consists of multiple phases. First, multiple images or data points of the object or environment are collected from various angles or viewpoints using sensors or cameras. The photos or data points are then analysed with algorithms to extract features and generate a point cloud or mesh that depicts the shape and geometry of the item or environment. To remove noise and mistakes, the point cloud or mesh is improved and cleaned up using mathematical techniques such as filtering and smoothing. Finally, textures and colours can be added to the 3D model to make it look more realistic and detailed. The basic idea behind 3D reconstruction is shown in the Fig. 3.9 where images of an object are captured from different views and their features are extracted, matched and triangulated to get the reconstruction of the object.

3.3.2 Mapping Module in SLAM

In SLAM (Simultaneous Localization and Mapping), the mapping module is in charge of creating a map of the environment while also estimating the robot's location within that map. A critical component of SLAM systems used in robotics and autonomous vehicles is the mapping module.

The mapping module in SLAM differs from 3D reconstruction in several important ways:

1. The purpose of 3D reconstruction is to generate a 3D model of an item or scene from a sequence of 2D photos, whereas the mapping module in SLAM generates a map of the environment the robot is investigating.

2. SLAM considers that the robot is moving and the environment is stationary, whereas 3D reconstruction normally assumes that the cameras are stationary and the object or scene is moving.

3. In SLAM, the mapping module must manage a variety of sensor data from devices like cameras, sonar sensors, and laser range finders and combine the data into a reliable map. On the other hand, 3D reconstruction often only functions with images captured by a single camera.

The mapping module must combine sensor data with robot motion data to determine the robot's position and orientation in 3D space in order to produce an environment map in SLAM. The mapping module then makes use of this data to create a 2D or 3D map of the area. SLAM's mapping module often employs probabilistic techniques to determine the location of the robot and the layout of the surrounding space. Particle filters, Kalman filters, and graph-based techniques like pose-graph optimisation are some of these algorithms.

Though both 3D reconstruction and SLAM involve building 3D models of environments, their objectives and methods differ. While 3D reconstruction is more concerned with producing precise 3D models of objects and scenes, SLAM is specifically created for robots to navigate and operate in uncharted environments.

3.3.3 Neural Radiance Fields

NeRF (Neural brightness Fields) is a contemporary computer vision and graphics technology that use deep neural networks to describe the brightness of a 3D scene as a continuous function. NeRF-based mapping is an extension of NeRF that tries to construct a 3D map of an environment using the same continuous function.

Recent advances in computer vision have resulted in novel approaches for 3D scene reconstruction, one of which includes directly mapping from 3D spatial coordinates to an implicit representation of the form using a multi-layer perceptron (MLP). This method uses machine learning to automatically learn the best function to represent the scene, allowing the model to take in a fixed-width vector of 3D coordinates and generate an RGB and density value along a ray to composite the ray and generate a final pixel value for image prediction. The model then compares its predictions to the original photos and computes a loss value, which it uses to alter its parameters and

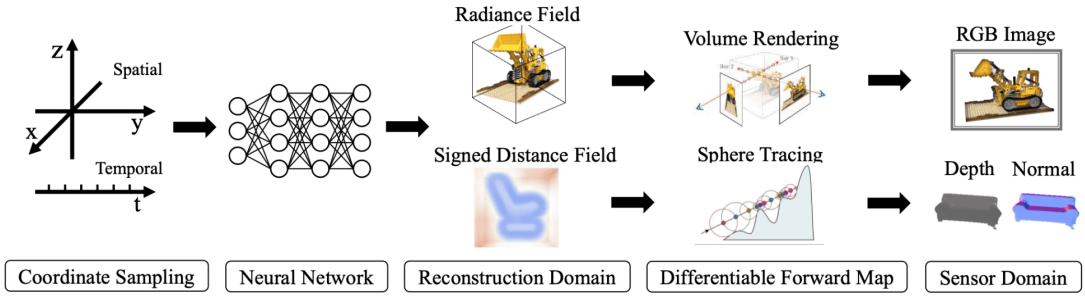

Figure 3.10: NeRF algorithm

generate a new iteration of the model with a smaller loss. MLPs have also been used in other areas of graphics, such as mapping low-dimensional coordinates to colours and representing textured materials and indirect illumination values.

Formulation

We emit rays at each pixel and sample at various timesteps while sampling coordinates from original pictures, a technique known as ray marching. Each sample point is identified by its geographical position, colour, and volume density. These are the neural field's inputs.

In the Neural Radiance Fields (NeRF) approach, a ray is described as a function of its origin point o, direction d, and its sampled points at different timesteps t. This function can be mathematically represented as

$$r(t) = o + td$$

The volume density and colour of the ray are both determined by the ray itself, which can be denoted as $\sigma(r(t))$ and $c(r(t))$ respectively. Now to map these rays back to the image we back-integrate the rays to obtain the pixel corresponding to the ray.

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t), d)dt$$

t_n and t_f are the bounds of the rays and $T(t)$ is the transmittance. In the context of NeRF, "transmittance" is a term used to measure how far a ray can travel through 3D space to reach a specific point, and it plays a critical role in the rendering process. This is because the amount of light that passes through a point in 3D space is determined by the transmittance value. In other words, transmittance regulates the amount of light that is absorbed or scattered as it traverses through the 3D scene, impacting the color and brightness of the final image. It is defined by

$$T(t) = \exp\left(-\int_{t_n}^{t_f} \sigma(\mathbf{r}(s))ds\right)$$

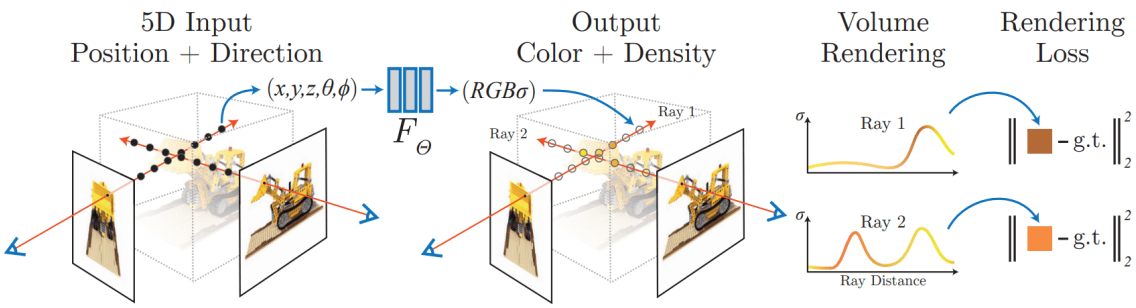

Figure 3.11: NeRF Training

3.3.4 Ray Marching

In order to generate the rays needed for NeRF you will need:

- Get the camera pose of each training image. It is represented by a 3D position vector + a 3D rotation representation (rotation matrix or quaternions usually) The calibration matrix K of the camera. It will provide the information on how to project pixels into the 3D scene.
- Compute rays for each pixels in the r from the camera pose, the pixel coordinates and K. This is done using Ray Marching/Ray tracing.

Our task consists of creating a primary ray for each pixel of the frame. This can easily be done by tracing a line starting at the camera's origin and passing through the middle of each pixel. We can express this line in the form of a ray whose origin is the camera's origin and whose direction is the vector from the camera's origin to the pixel center.

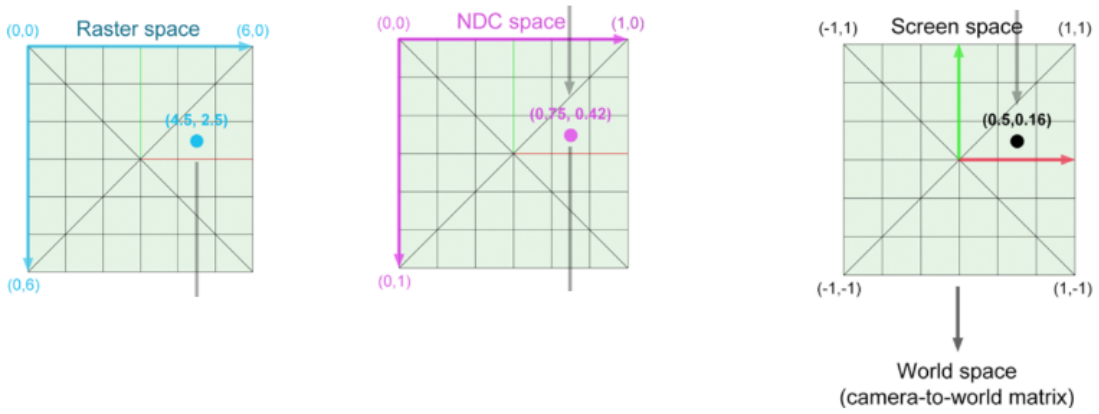

Figure 3.12: Representation of different coordinate spaces

There are 3 steps involved in the process:

- Raster space to NDC Space to screen space (world space)

$$PixelNDC_x = \frac{Pixel_x + 0.5}{ImageWidth}$$

$$PixelNDC_y = \frac{Pixel_y + 0.5}{ImageHeight}$$

$$PixelScreen_x = 2 * PixelNDCx - 1$$

$$PixelScreen_y = 2 * PoxelNDC_y - 1$$

- Transform each pixel to world space to camera space. Here αrefers to the viewing angle.

$$ImageAspectRatio = \frac{ImageWidth}{ImageHeight}$$

$$PixelCamera_x = (2 * PixelScreen_x - 1) * ImageAspectRatio * \tan\left(\frac{\alpha}{2}\right)$$

$$PixelCamera_y = (1 - 2 * PixelScreen_y) * \tan\left(\frac{\alpha}{2}\right)$$

- Find the direction vector from the camera origin to each pixel in the camera space.

$$P_{CameraSpace} = (PixelCamera_x, PixelCamera_y, -1$$

3.3.5 Comparision

	Method	Map Density	Map Type	Indoors			Outdoors			Loop Closure
				fr1/Desks	*fr2/xyz*	*fr3/Office*	*Kitti(09)*	*MH03*	*V103*	
RGBD-SLAM	Feature-Based	Dense	Both	2.3	0.8	3.2	8.3	2.8	4.8	Yes
ORB-SLAM2	Feature-Based	Dense	Both	1.6	0.4	1.0	8.3	2.8	4.8	Yes
LSD-SLAM	Direct	Semi-Dense	Both	10.5	LT	LT	10.5	LT	LT	Yes
PTAM	Feature-Based	Sparse	Indoors	LT	2.6	LT	×	×	×	No
iMAP	NeRF	Dense	Indoors	4.9	2.0	5.8	×	×	×	No
Nice-SLAM	NeRF	Dense	Indoors	2.7	1.8	3.0	×	×	×	No
MeSLAM	NeRF	Dense	Indoors	6.0	6.54	7.5	×	×	×	No
DTAM	Direct	Dense	Indoors	2.7	0.4	2.5	×	×	×	No
BAD-SLAM	Direct	Dense	Indoors	1.7	1.1	1.7	×	×	×	Yes
DSO	Direct	Sparse	Outdoors	×	×	×	9.0	17.2	2.5	No
DVSO	Deep Learning	Dense	Outdoors	×	×	×	8.3	–	–	Yes
D3VO	Deep Learning	Dense	Outdoors	×	×	×	7.8	8.0	11.0	No

TABLE 3.1: Tracking Accuracy in 3D maps for different SLAM approaches.

The table gives various approach details mainly the **Absolute Trajectory Error (ATE) as Root Mean Square Error(RMSE) in (cm)** with ground truth. The comparative study takes a reference to [13] and official results cited in individual approach papers. For indoors TUM RGB-D Dataset is used for evaluation and for outdoors KITTI Benchmark sequence 09 and EuRoC MAV dataset(MH03, V103). $'\times'$ means that the algorithm is not made to work in the respective type of mapping environment. 'LT' means at some point tracking was lost leading to a large section of the dataset being unmapped.

Chapter 4

Proposed Framework

4.1 Overview

The following section gives an overview of the framework/pipeline that is designed to create NeRF based 3D mapping.

We make use of raw rgb images captured from a realsense camera as input to the network and train a NeRF model to learn the 3D map of the environment in the form of depth radiance and color for each 3D point in the environment.

The framework consists of the following stages:

- **Camera Calibration**: Before using the images we have calibrated the camera to know its intrinsic parameters such as focal length, scale etc.
- **Pose Estimation**: Before mapping is done it is essential to localize the camera frames in the environment i.e. estimate its pose in the form of translation and rotation. The localization is done using visual odometry performed on a series of images obtained from the video recorded from Turtlebot.
- **Feature Map Generation**: This block consists of generation of surface normal maps and uncertainity map for each image in the dataset as an additional input provided to the NeRF network.

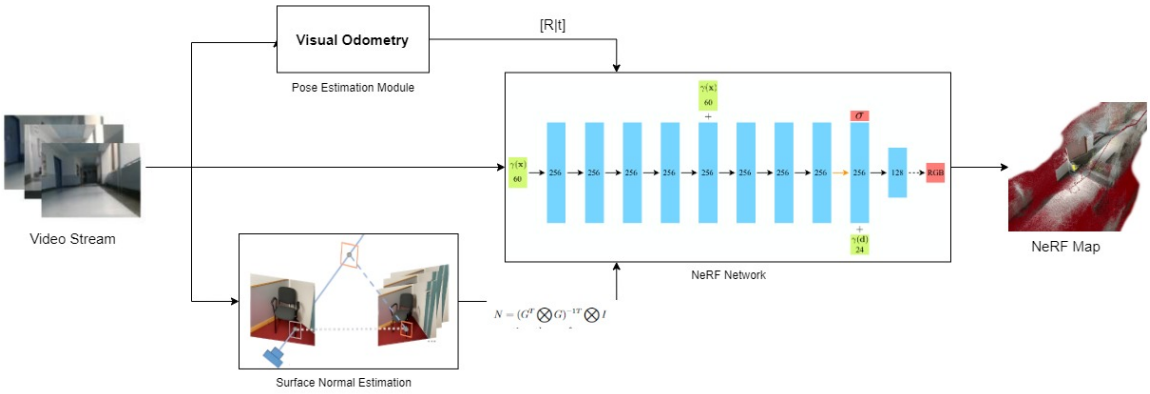

Figure 4.1: NeRF based Mapping Architecture

- **NeRF Architecture**: The NeRF architecture takes as input 5D parameters in the form of spatial location(x, y, z) and viewing direction(θ,φ) as input to the network along with the surfel maps and outputs 3D points with volume density depth and color for each point in 3D.
- **Volume Rendering**: In this step the estimated 3D model is queried to render an image. This is the part of novel view synthesis.
- **Loss Computation**: Here the rendered image is compared with ground truth image for supervision and backtracking using L2 Loss.
- **Evaluation Metric**: At this stage we evaluate the poses computed before map generation to ensure the accuracy of mapping.

4.2 Data Generation

The dataset considered for the project were both prebuilt as well custom made dataset.

- Prebuilt Dataset: We used the scannet dataset for evaluation as shown in results.
- Custom Dataset: The dataset was collected from the department in the form of a video. The dataset can be considered in a semi-outdoor dataset.

4.3 Camera Calibration

We have made use of the Chekerboaord method for camera calibration that involves capturing images of a planar checkerboard pattern from different viewpoints and using the known geometry of the checkerboard to estimate the camera parameters.The calibration matrix obtained f contains the intrinsic and extrinsic parameters of the camera. The intrinsic parameters include the focal length (fx, fy), principal point (cx, cy), and distortion coefficients (k1, k2, p1, p2, k3, ...).

$$s * [u; v; 1] = K * [R|t] * [X; Y; Z; 1]$$

Where K is the calibration matrix given in the form

$$K = \begin{bmatrix} F_x & \gamma & c_x \\ 0 & F_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

4.4 Pose Estimation

Once the camera has been calibrated the calibration matrix along with the iamges has been given as an input the the visual odometry algorithm that estimates the pose in the form of translation [tx, ty,tz] and rotation [rx, ry, rz]. We have used

the fast algorithm for feature detection adn 2D to 2D matching method for the pose estimation algorithm The rotation matrix are obtained form the fundamental matrix which encodes the relative orientation of two camera frames for a calibrated camera K. given by:

$$F = K'^{-T} R' S_b R''^{T} K''^{-1}$$

The value for F is obtained by applying the coplanarity constraint

$$x_n''^{T} F x_n' = 0$$

or

$$\begin{bmatrix} x_n' & y_n' & 1 \end{bmatrix} F_{(3x3)} \begin{bmatrix} x_n'' \\ y_n'' \\ 1 \end{bmatrix} = 0$$

and solving it using Singular value decomposition SVD to get a solution for F given known points in space x1,x2... in a

$$\begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{bmatrix} F = 0 such that$$

$$a_i = x_n'' \bigotimes x_n'$$

4.5 Feature Map Generation

Figure 4.2: Raw Image, Surface Normals, Depth Map

Figure 4.3: Raw Image, Surface Normals, Depth Map

The feature map that is used as a helper input to the network is the surface normal . Surface normals are vectors that are at each point perpendicular to a surface. They reveal details about the surface's direction and contour.The cross product of two

vectors on an object's surface is used to determine surface normals. The first vector is a line joining two nearby locations, while the second vector is perpendicular to the plane those two points form. At the point of interest, the resulting vector is perpendicular to the surface. One can determine the shape of surfaces and objects in a scene by computing the surface normals for each point in a point cloud.

The mathematical expression for computing surface normals from an image is:

$$N = (G^T \bigotimes G)^{-1T} \bigotimes I$$

where N is a vector representing the surface normal at a given pixel, G is a matrix representing the gradient of the surface at that pixel in each of the lighting directions, and I is a vector representing the intensity values of the pixel under each of the lighting conditions.

The matrix G is typically defined as:

$$G = \begin{bmatrix} l_1x & l_1y & l_1z \\ l_2x & l_2y & l_2z \\ \vdots & \vdots & \vdots \\ l_nx & l_ny & l_nz \end{bmatrix}$$

where lix, liy, and liz are the components of the light direction vector for the ith image.

4.6 NeRF Architecture

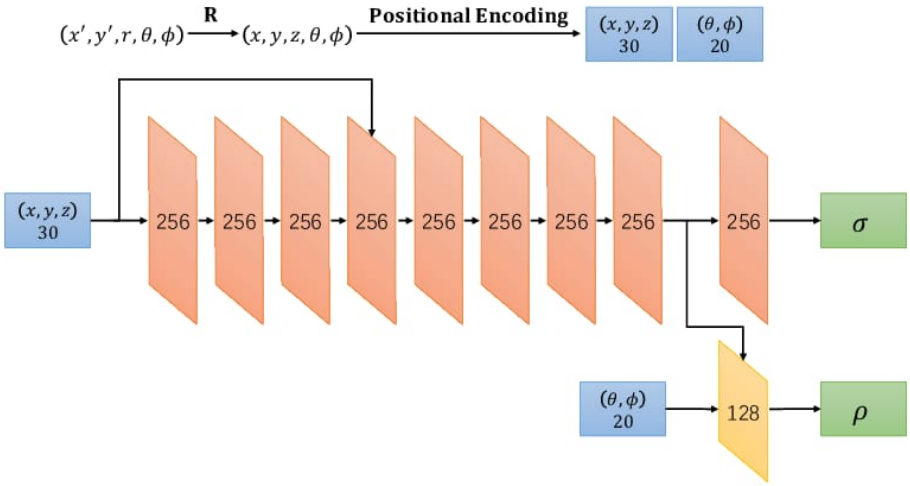

Figure 4.4: NeRF Architecture

Output is a 3D scene giving its emitted color c = (r, g, b) and volume density σ per pixel per image as its output given known camera parameters.

Hence the model makes a mapping FΘ: (x, d)→(c,σ) based on the rendering equation, such that expected color C(r) of camera ray r(t) = o + td with near and far bounds tn and tr.

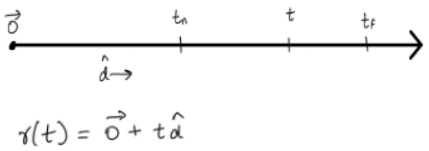

Figure 4.5: Ray marching

$$C(r) = \int_{t_f}^{t_n} T(t)\sigma(r(t))c(r(t), d)\, dt, WhereT(t) = exp(-\int_{t_f}^{t_n} \sigma(r(s))\, ds)$$

Here T(x) is the accumulated transmittance along the ray from tn and tf. The light reflecting properties of the object (density) is denoted by the volume density σ(r(t)),or the differential probability of a ray terminating at an infinitesimal particle at location x=r(t). c(r(t),d) refers to the total light that the point at t of r(t) sends in the direction d. The conversion of every pixel location into viewing direction(θ,φ) is done by converting cartesian coordinates in global frame to spherical coordinate

$$\rho = t_x^2 + t_y^2 + t_z^2$$

$$tan\theta = \frac{t_x}{t_y}$$

$$\phi = arccos(\frac{t_z}{\rho})$$

The values of colour and volume are clubbed with each pixel location in 3D to render the 3D map of the environment.

4.7 Volume Rendering

In this step the estimated 3D model is queried to render an image using a combination of input values(x,y,z,θ,φ) and obtained corresponding estimates of colour and volume density (RGB, σ) to render the 3D map for an image. Here we may turn the volume density profile along a ray into a pixel value. Then we repeat the process for every pixel in the image to get a complete rendered image.

The higher the volume density of the point, the higher the probability of the ray stopping at that point, and thus the probability that the RGB value of that point will have a significant impact on the final RGB file of the rendered pixel. The rendering equation is same as used while formulation.

4.8 Loss Computation

A set of rendered images (one per pose) H,W_n and a set of ground truth images (one per pose) H,W_{ngt} as input is used to compute the L2 loss between them. Thus we

compare the pixels of the rendered image from the learned 3D map with the pixels of the ground truth image at a known pose to calculate the loss. It is given by:

$$L2 = \min_{\omega} \sum_{i} (||render^{(i)}(F_{\omega}) - I^{i}_{gt}||)^2$$

4.9 Evaluation Metrics

The evaluation metrix is mainly devised for the accuracy of the localization of frames done i.e the accuracy of the poses estimated. Depending on the quality of the poses the quality of the reconstruction depends. Although there are other factors that also affect the quality of reconstruction such as illumination, resolution of images, relative overlap percentage of respective frames etc, these factors cannot be controlled and changed. However the algorithm for pose detection needs to be as accurate as possible. There are 2 metric values for evaluation of poses int he form of trajectory namely:

- Absolute Tracking Error: ATE is the Euclidean distance between the ground truth camera pose and the estimated camera pose at each time step. It is calculated as:

  $$ATE = \sqrt{1/N} * \sum_{n=1} (||T_g t_i - T_e s t_i||)^2$$

  where Tgt and Test are ground truth observed from imu sensor and estimated.

- Root mean squared error: RMSE is a measure of the average deviation of the estimated trajectory from the ground truth trajectory, and is calculated as the square root of the mean of the squared errors between the two trajectories. It is calculated as:

Chapter 5

Results and Discussions

5.1 Software Setup

5.1.1 ROS Neotic

Figure 5.1: ROS Neotic

ROS (Robot Operating System): ROS is an open-source software framework commonly used to develop and control robots. It provides a collection of libraries, tools, and conventions to simplify creating complex robot behaviour. Neotic is the version of ROS that was used for data collection.

5.1.2 RViz

Figure 5.2: RViz

Rviz, short for ROS visualization, is a robust 3D visualization tool designed specifically for ROS. It enables users to visualize the simulated robot model, log sensor data from the robot's sensors, and replay the recorded sensor data.

5.1.3 TurtleBot 2

Figure 5.3: TurtleBot 2

TurtleBot is a popular open-source, low-cost mobile robot platform aimed for teaching, research, and hobbyists. It is built on the open-source Robot Operating System (ROS) and may be used to investigate several areas of robotics such as sensing, perception, control, and navigation. This robot was used for data generation.

5.2 Experimental Setup

In this experiment, we used an RTX 3090 GPU for training a NeRF model. The GPU was equipped with 24 GB of GDDR6X memory and a core clock speed of 1.40 GHz.

We trained the NeRF model using 35 frames of input data, which consisted of a set of images captured using a Intel RealSense camera with a resolution of 640x480 pixels. The camera was positioned at a fixed and mounted on TurtleBot 2.

The input data was preprocessed and fed into the NeRF model, which was implemented using PyTorch deep learning framework. The training process was run on the RTX 3090 GPU using the Adam optimizer with a learning rate of 1e-4.

The experiment was conducted on a high-performance computing cluster with a dedicated GPU node. The results of the experiment were evaluated using various quantitative metrics, including the mean squared error and the structural similarity index measure.

5.3 Results

5.3.1 NeRF model output using Standard Dataset

The NeRF model was created using the publicly available fox dataset. Initially, the model was trained on the images within the dataset and subsequently evaluated by providing the model with input in the form of a viewing direction and ray.

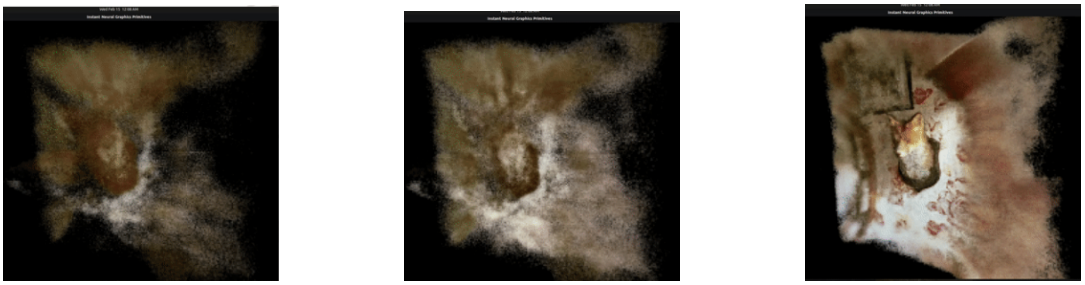

Figure 5.4: Training the NeRF model using images from a dataset.

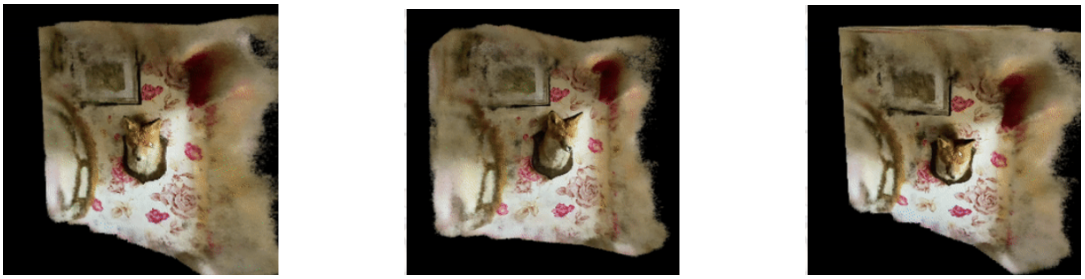

Figure 5.5: Testing the trained NeRF model by querying the colour and distance from a particular viewing angle

Figure 5.6: Testing the trained NeRF model from a custom dataset of the mouse.

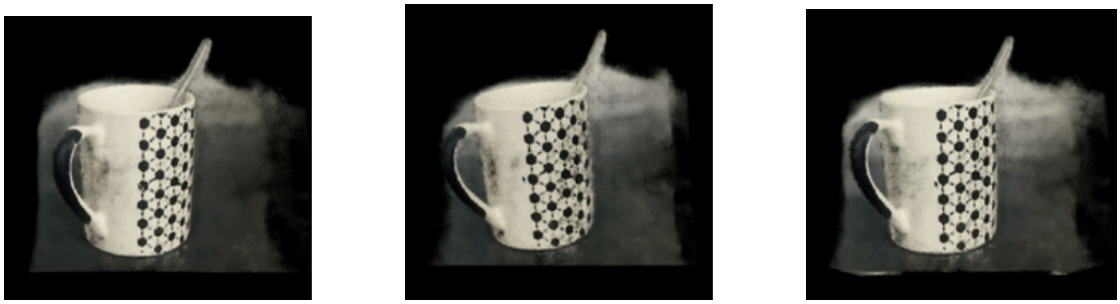

Figure 5.7: Testing the trained NeRF model from a custom dataset of mug

5.3.2 NeRF output using a custom-made dataset

5.3.3 Visual Odometry

The visual odometry algorithm mentioned in Section 3.2.3 is used to generate the trajectory on our custom dataset.

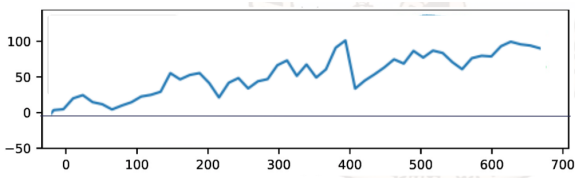

Figure 5.8: Visual Odometry Result - 1 for straight track using FAST Features

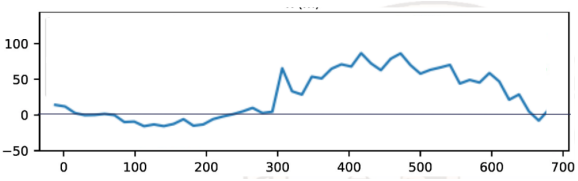

Figure 5.9: Visual Odometry Result - 2 for the straight track using SIFT Features

	Max Deviation (cm)	**Mean Deviation (cm)**	**Accuracy**
Trajectory - 1	102.3	60.22	84.86
Trajectory - 2	80.82	50.45	90.23

TABLE 5.1: Visual Odometry Trajectory Metrics

5.3.4 NeRF Map Generation on Standard Dataset

The map of the room was generated using the NeRF network which was trained using the ScanNet dataset available online.

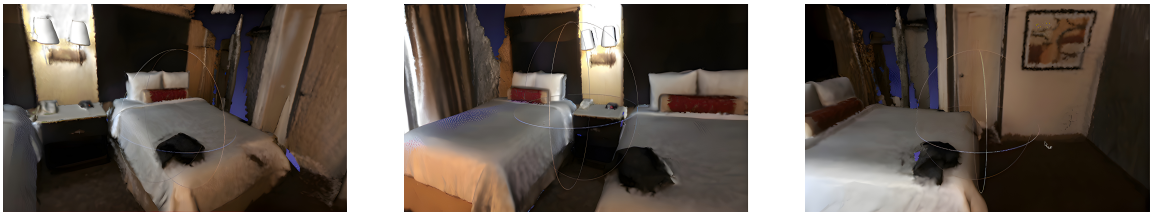

Figure 5.10: NeRF Map using ScanNet Dataset

5.3.5 NeRF Map Generation on custom-made dataset

The below results are of map that was generated using NeRF network which was trained on custom dataset.

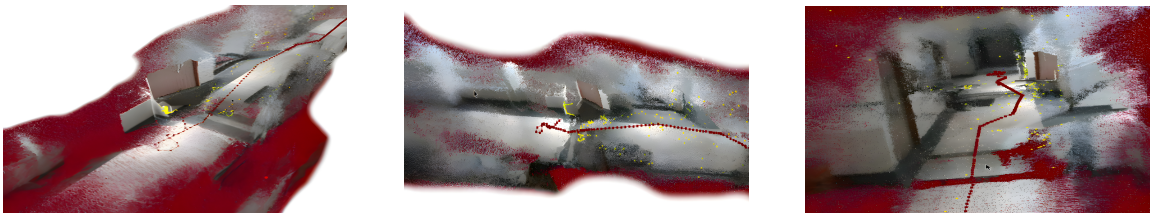

Figure 5.11: NeRF Map using ScanNet Dataset

Chapter 6

Summary and Conclusions

We present a NeRF based mapping method to improve the challenges faced by standard point cloud based maps used in localization and slam navigation. We have made use of raw RGB images as input to the network. The proposed method is an amalgamation of tradition approaches and learning based methods. The pose estimation has been done using visual odometry which is a classical method for visual localization while we have made use of Neural Radiance Fields which is a deep learning based method to perform mapping of the environment.

6.0.1 Future Works

- The NeRF based 3D map can be used in the future for more efficient path planning and navigation.
- Furthurmore since the proposed network has the capability to predict occluded 3D points it can be used to obtain denser maps which are essential for accurate navigation.
- The novel view synthesis capability of NeRFs opens the door for precise robot re-localization and loop closure.

Bibliography

[1] Andrew J Davison et al. "MonoSLAM: Real-time single camera SLAM". In: *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007), pp. 1052–1067.

[2] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[3] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*. Springer. 2014, pp. 834–849.

[4] Giorgio Grisetti et al. "g2o: A general framework for (hyper) graph optimization". In: *Proceedings of the IEEE international conference on robotics and automation (ICRA), Shanghai, China*. 2011, pp. 9–13.

[5] Evgenii Kruzhkov et al. "MeSLAM: Memory Efficient SLAM based on Neural Fields". In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2022, pp. 430–435.

[6] Ben Mildenhall et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[7] Michael Montemerlo et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *Aaai/iaai* 593598 (2002).

[8] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[9] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. "DTAM: Dense tracking and mapping in real-time". In: *2011 international conference on computer vision*. IEEE. 2011, pp. 2320–2327.

[10] Taihú Pire et al. "S-PTAM: Stereo parallel tracking and mapping". In: *Robotics and Autonomous Systems* 93 (2017), pp. 27–42.

[11] Davide Scaramuzza and Friedrich Fraundorfer. "Visual Odometry [Tutorial]". In: *IEEE Robot. Automat. Mag.* 18 (Dec. 2011), pp. 80–92. DOI: 10.1109/MRA.2011.943233.

[12] Thomas Schops, Torsten Sattler, and Marc Pollefeys. “Bad slam: Bundle adjusted direct rgb-d slam”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 134–144.

[13] Edgar Sucar et al. “iMAP: Implicit mapping and positioning in real-time”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6229–6238.

[14] Matthew Tancik et al. “Block-nerf: Scalable large scene neural view synthesis”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 8248–8258.

[15] Qiangeng Xu et al. “Point-nerf: Point-based neural radiance fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5438–5448.

[16] Lin Yen-Chen et al. “inerf: Inverting neural radiance fields for pose estimation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1323–1330.

[17] Z. Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[18] Zihan Zhu et al. “Nice-slam: Neural implicit scalable encoding for slam”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12786–12796.