Efficient Deep Learning-Based Semantic Mapping Approach Using Monocular Vision for Resource-Limited Mobile Robots

Aditya Singh[1,2*], Raghav Narula[2,3], Hatem A. Rashwan[1], Mohamed Abdel-Nasser[4], Domenec Puig[1] and G C Nandi[2]

[1]Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, Spain, Tarragona, Spain.
[2]Center of Intelligent Robots, Indian Institute of Information Technology, Allahabad, India.
[3]Thapar Institute of Engineering and Technology, India.
[4]Department of Electrical Engineering, Aswan University, Aswan, Egypt.

*Corresponding author(s). E-mail(s): aditya.singh@urv.cat;
Contributing authors: rnarula_be19@thapar.edu; hatem.abdellatif@urv.cat; mohamed.abdelnasser@aswu.edu.eg; domenec.puig@urv.cat; gcnandi@iiita.ac.in;

Abstract

Semantic mapping is still challenging for household collaborative robots. Deep learning models have proved their capability to extract semantics from the scene and learn robot odometry. For interfacing semantic information with robot odometry, existing approaches extract both semantics and robot odometry separately and then integrate them using fusion techniques. Such approaches face many issues while integration, and the mapping procedure requires a lot of memory and resources to process the information. In an attempt to produce accurate semantic mapping with resource-limited devices, this paper proposes an efficient deep learning-based model to simultaneously estimate robot odometry by using monocular sequence frames and detecting

objects in the frames. The proposed model includes two main components: using a YOLOv3 object detector as a backbone and a convolutional long-short term (Conv-LSTM) recurrent neural network to model the changes in camera pose. The unique advantage of the proposed model is that it boycotts the need for data association and the requirement of multi-sensor fusion. We conducted the experiments on a LoCoBot robot in a laboratory environment, attaining satisfactory results with such limited computational resources. Additionally, we tested the proposed method on the Kitti dataset, reaching an average test loss of 15.93 on various sequences. The experiments are documented in this video https://www.youtube.com/watch?v=hnmqwxpaTEw.

Keywords: Visual Odometry, Object Detection, Household Robots, Mapping, Agglomerative Clustering

1 Introduction

Robots are becoming increasingly important for human-centric tasks in the home. Robots can help us in a variety of ways, from elderly care to rubbish picking. Semantic understanding of the environment is required to improve human interaction and comprehend human commands. Semantic maps can express the environment meaningfully to robots by extracting the common-sense knowledge from the scene and combining it with object and place classification [1]. Indoor semantic maps of an environment have been created using several ways [10–12]. The majority of existing approaches have combined many sensors, such as Lidar, infrared sensors, and cameras, and superimposed semantic information on top of them. Although these methods have demonstrated their worth in terms of performance, their implementation in real-world robots is typically expensive and suffers from the difficulty of data association.

Recent object detection algorithms like YOLO (You Only Look Once) [2], SSD (Single Shot Detector) [3], and their advancements have shown a substantial efficacy in extracting semantics from the scene in real-time, thanks to advances in deep learning. As a result, utilizing deep learning-based object detection algorithms in mapping problems enhances the performance and efficiency of constructing semantic maps dramatically. Furthermore, deep learning algorithms have been shown to be effective in the learning of visual odometry [25, 33]. Deep convolutional neural networks (CNNs)-based feature extractors were utilized in certain studies, such as [25, 33], followed by a recurrent neural network. The stereo vision correspondences for creating ground truths or ground truth measured by another sensor are both used in the learning process. The use of CNNs in all models is confined to feature extraction, and it does not serve any mode of application. Besides, existing approaches for integrating semantic data with robot odometry extract both semantics and robot odometry independently and then combine them using fusion techniques. Such techniques confront numerous challenges during integration, and the mapping

operation necessitates a significant amount of memory and resources to handle the data. In order to provide reliable semantic mapping on devices with limited resources, this paper presents an efficient deep learning-based approach to simultaneously estimate robot odometry and detect objects in monocular sequence frames.

The proposed semantic mapping model has two key components: an object detector as a backbone and a convolutional long-short term (Conv-LSTM) recurrent neural network to model changes in camera pose. The proposed model has the advantage of avoiding the necessity for data association and the requirement for multi-sensor fusion. Specifically, we utilize an object detector for feature extraction and extraction of semantics from the scene. We have used a YOLOv3 object detector [27] because of its fast detection rate and easy to integrate architecture. According to [50], Yolov3 architecture results better than other claimed architectures for mAP (mean Average Precision) versus speed combination, which supports its use for this application. The output of the first detection layer of YOLOv3 serves as the input to Conv-LSTM. We have used Conv-LSTM blocks instead of normal LSTM because the Conv-LSTM layer proved effective while handling the feature map output. The final output of the architecture is the change in odometry values between sequence frames and bounding box coordinates same as YOLOv3 for the objects present in each frame. Furthermore, utilizing the semantic information and the corresponding odometry value, a mapping technique is employed to create a semantic map. It calculates the distance between detected objects and the robot using a geometrical relationship between the camera plane and the environment. The distance calculated from each frame is then collated using agglomerative clustering [43].

The following is how the rest of the article is organized: Section 2 examines existing semantic mapping and monocular visual odometry approaches, as well as their limitations in relation to the current challenge. We discussed our contribution to the problem of semantic mapping in Section 2.3. The methodology is detailed in Section 3, which includes a description of both private and public datasets, as well as the entire approach and network design. In Section 4, the results of various monocular odometry and mapping tests are discussed. Section 5 summarizes the study and discusses the results, constraints, and future scope of the problem.

2 Related Work

The existing machine learning-based and feature extraction-based semantic mapping and visual odometry algorithms are presented and discussed in this section.

2.1 Semantic Mapping

At various scales, the SLAM problem (simultaneous localization and mapping) is tackled utilizing a variety of machine learning and feature extraction

approach. It reaches a point of maturity where detailed city-level maps may be created [4–6]. The semantic description of scenes and the objects included in scenes became a priority for mapping as the demand for human-centric operations grew. This mapping is known as semantic mapping, and it is based on ontology to define concepts, relations in maps, and objects in order to rebuild a SLAM map that provides the scene geometry and object locations for the moving robot.

The process of assigning semantic meaning (object categories, identities, actions, and so on) to the items being mapped is known as semantic mapping. Reconstruction and segmentation of the reconstructed map were the first efforts in semantic mapping. The base for semantics is either object detection or semantic segmentation. It is dependent on the method that is most appropriate for the system. Because the amount of segmentation classes increases the parameters of the neural network, segmentation models are heavier than detection models. In terms of performance and network architecture, both strategies are evolving with time. Traditional SLAM tools or some advances are visible for mapping. Semantic mapping can be done using a variety of approaches.

Pham et al. [7] firstly reconstructed a dense 3D model from RGB-D images using Kinect Fusion then assigned each 3D point to a semantic label using a hierarchical Conditional Random Fields (CRF) model. Mozos et al. [8] generated semantic maps with range sensors into functional spaces using a Hidden Markov Model (HMM). They demonstrated that the semantic information obtained could be used to convert into a topological map. Vineet et al. [9] proposed an online dense reconstruction method that solves the semantic labeling problem as a densely connected CRF. Kundu et al. [34] derived a CRF model over a 3D space that jointly infers each voxel's semantic category and occupancy. The CRF model is used to exploit the complex information for contextual cues in a 3D space. It lacks in the case of multi-scale problems, and because it is based on a descriptive process, the semantic labels are not rich. The resulting mapping is very irregular and disorganized.

In addition, Sunderhauf et al. [10] used an RGB-D sensor for creating depth maps, and an ORB-SLAM was then used for mapping and localization purposes. In [10], the SSD [3] detector was used to detect objects in the scene, and then a 3D point cloud segmentation was used to impose semantic information on the point cloud representation. The performance of the SSD model was compared to other available algorithms and found most appropriate for the object detection purpose. A Laser scanner with an Orbecc Camera was proposed in [11] for semantic mapping, and the YOLO network [2] was used for localizing the objects in the scene. Besides, a laser scan matcher technique was used for metric mapping. In both mapping tasks, the process of the localization and extraction of semantics were separately achieved. In [11], the division of different spaces (rooms) was conducted by using predefined knowledge as well as a lag while integrating two hardware results was created. A panoptic fusion was performed in [12] by using an RGB-D sensor for the volumetric

identification and localization of the objects in the scene. [12] used a Scan-Net architecture to process the RGB and depth images simultaneously.

The above-mentioned works *main limitation* is their reliance on multiple hardware, which causes difficulty during data association. The use of these techniques necessitates the inclusion of a large amount of memory (at a significant expense) to the robot setup. Unlike previous methods, the proposed method reduces reliance on specific hardware while achieving sufficient semantic mapping results with such limited computational resources.

2.2 Monocular Visual Odometry

Visual Odometry (VO) is employed to predict the camera's ego-motion using a sequence of frames. The frame-to-frame correspondences provide constraints for camera pose and map estimation, and camera intrinsic parameters are used to relate the frame correspondences. Visual odometry is basically solved in three different ways: Sparse Feature Approach, Direct Methods, and Deep Learning Methods.

Nistér et al. [13] proposed a method that leveraged stereo vision to solve the VO problem. However, the accuracy of the algorithm is reduced over time. Then, other researchers [39, 40] presented Visual Simultaneous Localization and Mapping (VSLAM) as a better and more accurate mapping and localization. For instance, the early monocular vision was achieved by using filter-based algorithms [14–17] like Extended Kalman Filter (EKF). The EKF uses the idea of state vectors for storing the poses and the 3D coordinates to store and make a map from the poses. Though accurate, the EKF system has the problem of uncertainty caused by complexity and linearization. Consequently, [18–20] proposed to many extents the Unscented Kalman Filter (UKF) improve upon the EKF; however, UKF suffers from the problem of computation complexity, which results in decreasing usage in real-time. Increasing upon the usage in real-time, the authors of [21] developed a PTAM (Parallel tracking and mapping) algorithm that extracts key-frames. In turn, in [22, 45], the authors attempted to add edge features to the map and exploit their resilience to motion blur to improve tracking under fast motion.

The authors of [22] also implemented a very simple inter-frame rotation estimator to aid tracking when the camera is rapidly panning, which provided some very compelling results when the camera moves quickly. Further improving upon the previous works, ORB-SLAM (Oriented FAST and rotated Brief) was developed in [23] through a method that selects survival of the fittest strategy that determines the points and key-frames of the scene reconstruction leading to a compact yet trackable map. The whole SLAM process is divided into three main threads localization, mapping, and loop closing. The method uses ORB features for these three tasks and introduces the concept of Essential Graph to help in the correction process of loop closing.

Although feature-based approaches are often practical and accurate, they can have certain drawbacks, particularly when there is a pure rotation, which makes extracting key points and calculating descriptors highly

time-consuming. All information other than feature points is ignored when employing feature points. The majority of images comprise thousands of pixels but just a few hundred feature points, resulting in a significant loss of information. Researchers presented a direct solution based on pixels rather than key points in an image to address the aforementioned shortcomings. A direct technique is employed in the technology known as DTAM (Dense Tracking and Mapping In Real Life) [24]. In terms of hardware requirements, the direct technique is inconvenient of devices of limited resources.

Deep learning models are used in a variety of ways to learn the frame correspondence. In [25], the use of CNN, optical flow, and LSTM (long-short term memory) [35] combined to give out some results that were acceptable and very close to ground truth. Cementing on that, Magic VO [38] also proposed a novel architecture of combining CNN with Bi-LSTM [37] to more precise results. Furthermore, various unsupervised methods are also available for the estimation of camera ego-motion or robot odometry. For instance, DeepVO proposed in [33] is based on LSTM cells and is used for learning VO on the Kitti dataset. The *main limitation* of such learning models is the unavailability of semantic features. For extracting semantics, a separate model is needed, which again creates a mixing issue.

In addition, Pandey et al. [47] have proposed a convolution-RNN network for learning odometry change using sequential RGB images. The features are first extracted using CNNs and then fed into an RNN network which learns the sequential dependence on each other. The RNN network exploits the optical flow information extracted by the convolution network to learn odometry. In [48], a hybrid deep learning-based model is proposed, which uses deep learning capability for image processing and combines it with geometry-based pose localization. It uses optical flow and depth information to extract geometrical information. The *main limitation* of these models is that they work for only a single application (odometry) and the architecture of these models is inadequate for the limited computing resources of mobile robots due to their high computation cost.

2.3 Contribution

The above-mentioned approaches rely on complex feature extraction methods and a fusion scheme to combine the information. Such methods require different sensors and multiple information processing systems. Existing deep learning-based methods are heavy and unsuitable for devices of limited resources. To handle these issues, we propose a deep learning model for simultaneous learning visual odometry from a sequence of frames of a video and detecting objects in the scene. The proposed model uses the YOLOv3 model [27] as a primary network and takes feature maps from the detection layers into account. The YOLOv3 was chosen because it can extract reliable and necessary features from an input image at the detection level, and it is fast enough to complete the operation in real-time. The selection is based on a trade-off

between speed and availability of required features for the proposed odometry model. YOLOv3 achieves the predictions at three different scales, given by down-sampling the dimensions of the input image by a stride of 32, 16, 8, respectively. The outputs of the three scales with an image of 416×416 are: $13 \times 13 \times 255$, $26 \times 26 \times 255$, and $52 \times 52 \times 255$. We found that the $13 \times 13 \times 255$ is responsible for correctly representing the scene features. Thus, we will use the 13×13 feature map for estimating the pose change.

Over successive frames, features collected can characterize the motion and pose change in the scene as a time series. In such cases, an interesting approach is to use a model based on LSTM (Long-Short Term Memory) [35], which is a Recurrent Neural Network (RNN) [36] architecture. In this kind of architecture, the model passes the previous hidden state to the next step of the sequence. It holds information in prior data the network has seen before and uses it to make decisions based on the data order. Consequently, a Conv-LSTM [28] layer-based neural network is used for learning the pose change of the robot. It is a recurrent layer similar to LSTM [46], but internal matrix multiplications are exchanged with convolution operations. The learning is achieved in a supervised manner, and ground truth is used as a reference.

After getting the pose information and semantic information of the objects, geometrical modeling is used to locate the objects on the floor. The Manhattan world assumption proposed in [26] is used as a base for the calculation. The calculations from every scene being integrated correspond to the following frame calculation using a novel clustering approach. Finally, the semantic mapping process ends with a 2D map containing all the objects located on the floor. In the case of a household setup, the objects like chairs, refrigerators, Tables, Bed are considered ground objects and can be detected by the robot. The unique advantage of the proposed model is that it works in a flawless manner on the processor of LoCoBot robot, which has limited memory. The experiments are documented in this video https://www.youtube.com/watch?v=hnmqwxpaTEw.

In this work, the following contributions are listed:

- A fully data-driven semantic visual odometry algorithm is proposed for resource-limited devices. It is a deep learning-based algorithm for simultaneously estimating robot odometry and recognizing objects in monocular sequence frames. It contains two main components: using a YOLOv3 object detector as a backbone and a convolutional long-short term (Conv-LSTM) recurrent neural network to model the changes in camera pose.
- A geometry-based approach is proposed to establish the relationship between camera plane information to distance of objects in the real world. This results in the form of a 2D map with the location of all floor objects as points.
- A novel clustering method is proposed for integrating information between two consecutive frames. It results in recognizing the location of a particular object in two different frames by using the odometry values.

- An own private dataset is presented to investigate the performance of the proposed model for pose estimation and object detection.

3 Methodology

This section has discussed the proposed supervised learning approach for semantic mapping by using a sequence of monocular frames as input. It consists of the dataset collection process with a description of the robot hardware used for the experiment. Subsequently, a description of the architecture for object detection and pose change prediction is given, and the proposed model's training procedure is discussed. The model's performance is compared in terms of standard error metrics for object detection and visual odometry. We have proposed a mapping method for making a 2D map corresponding to an image frame for objects present in the image. It provides a map for objects located on the floor corresponding to the robot base using the identified bounding box coordinates. A clustering algorithm is discussed to integrate the map location of objects in different frames.

3.1 Problem Definition

To make a semantic map using the input of a robot mounted camera, we need to track the robot's location, localize the present objects corresponding to each frame with respect to the robot, and then merge them in a single map corresponding to a global address.

For localizing, we have three variables for the pose change of the robot. For a 2D planner movement, [X, Y, ω] are the variables for the robot in which two translation components (X and Y) shows the position on the robot on a 2D plane (in our case, floor), and one rotation component (ω) shows the change in orientation of the robot with reference to initial configuration. The dataset on which the pose estimation model is trained contains input images and corresponding pose values. Recurrent neural networks, i.e., LSTM with convolution (Conv-LSTM), are used to learn the pose variables. The input to the LSTM is a feature map extracted by the object detection model, YOLOv3. The Yolo model is pre-trained on the COCO dataset [44]. All the detected objects are firstly classified as a floor object or a non-floor object and all the floor objects are calculated for distance from the robot base. Based on the robot's location and objects detected for each pose, every object from the floor is being assigned a position on the map. All frame wise maps will be integrated using a clustering algorithm to avoid multiple occurrences of the same objects in different frames.

3.2 Dataset Description

Own-Private Dataset: The model is tested on two different datasets for visual odometry. A laboratory dataset recorded on a Locobot robot is used for

Fig. 1 Locobot Robot. It is used for recording of dataset and experiment purposes. An overhead camera with Depth and RGB semsor is available. Kobuki base is equipped with wheels and odometer

odometry and semantic mapping, and the standard KITTI dataset is used for comparing visual odometry performance.

In this paper, a Locobot robot is used for experiment purposes, as shown in Figure 1. Locobot is based on the PyRobot platform [42]. As shown in Figure 1, a fixed camera on the top of the robot is used for capturing the video. The camera module is an Intel Realsense with an RGB and a Depth Sensor [41]. A kobuki robot base is used for the mobility of the setup. The dataset comprises sequence-wise RGB images, corresponding depth images, and wheel odometry values. The odometry value has three values: two translation coordinates and one rotational coordinate, which is yaw motion. The odometry reading is recorded at a shallow speed to avoid slipping, and further, it is being corrected using the readings collected by an ORB-SLAM algorithm. The dataset consists of 13000 sequence RGB frames of image size 640×480. The sequences are split into training, validation, and testing sets as peruse. The images in the dataset include several objects like Chairs, Monitors, CPU Cabinets, and different occluded objects. We tried to create a household set-up in the recording of the dataset. In Table 1, a brief detail is available for different components used in the process of the data collection. Table 1 includes GPU, which is used for training of the model and RAM value has two values, one is used for training of the model and other is used for application on robot.

KITTI Dataset: KITTI proposed in [29] is a benchmark dataset for visual odometry, and it contains various scenarios for testing. In this paper, the KITTI dataset is used to generalise the proposed model for Visual odometry. The model is tested on four different sequences of the KITTI dataset. KITTI have six pose parameters including three rotation and three translation

Table 1 Hardware used in the semantic mapping process

Component	Hardware
Camera	RGB Camera of Intel Real Sense
Robot	LoCobot
CPU	Multi core processor
GPU	Tesla K80 with 2496 cores (for training only)
RAM	25 GB (training), and 8 GB (application)

parameters. We have selected only three parameters for making it compatible for our application. The chosen parameters include two transaltion and one rotation component. To make this selection worth, we have validated the selections by generating a continuous odometry using the values. In KITTI, the data is recorded by using a vehicle which used to perform 2D motion only so the selection of only three parameters is justified.

3.3 Architecture of the proposed model

The proposed network is built on the YOLO v3 model proposed in [27]. For pose estimation, an extension is achieved using the conv-LSTM layers introduced in [28] on the coarse detection layer of the YOLO v3 model. The selection of the coarse detection layer is based on empirical performance comparison, and the feature maps resulting from the coarse detection layer yield the best accuracy. The feature image output of the detection layer has a size of $13 \times 13 \times 255$, and it is converted to a 5D tensor in (samples, time_steps, channels, rows, columns), and it is then fed into the conv-LSTM network. As shown in Figure 2, the proposed network is a two-fold process. The network firstly uses a sequence of images and process them through the backbone of the YOLO v3 model, the DARKNET-53 model suggested in [27] for feature extraction. The DARKNET-53 model originally has 53 layer network trained on ImageNet. Afterwards, in the feature extraction part, there are two separate branches for object detection to localize the object in the scene and odometry for pose estimation.

Indeed, data explicit or implicit information fusion is indispensable for overcoming the syndrome of uncertainty in decision-making and problem-solving processes. Most existing approaches for semantic mapping are based on integrating both semantics and robot odometry based on explicit fusion techniques to match the information collected from diverse sources. Such techniques are limited by the available amount of memory and resources. In contrast, the proposed deep learning-based technique explicitly eliminates the need to fuse two separate pipelines for object localization and semantics extraction for robot odometry. The proposed deep-learning method, to-some-extent, applies a sort of implicit fusion in the feature space since the deep convolutional layers are trained with a large number of images to optimally learn powerful features (representation) for both object detection and robot odometry tasks, as shown in Figure 2 (i.e., Common Feature Space for object detection and odometry). Such common features that are shared across object detection

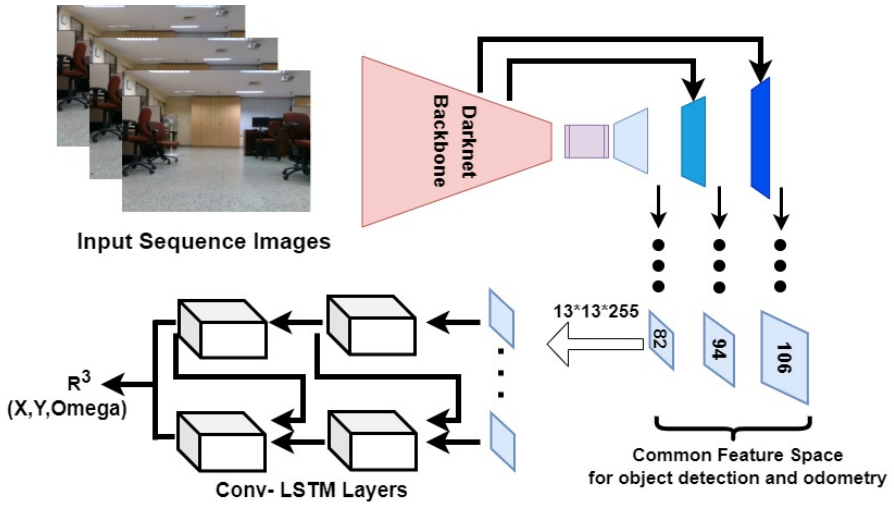

Fig. 2 Overview of the proposed architecture. Initially, the feature extraction part of the network is the same as that of YOLO v3. At detection level, 13×13×255 layer is taken as the input for the pose change estimation and pose change is learnt for the 2D motion, i.e., two translations and one rotation parameter. Common feature Space with 82,94, and 106 layer of YOLO v3 is shared with the task of odometry and object detection. These layers are combined and further used for object detection

and robot odometry tasks enhance performance relative to learning each task independently.

3.3.1 Object Detection

In YOLO v3, the network architecture boasts residual skip connections and upsampling. The most salient feature of YOLO v3 is that it makes the detection at three different scales (82, 94, 106 layers). YOLO is a fully convolutional network, and its eventual output is generated by applying a 1 x 1 kernel on a feature map. In YOLO v3, the detection is done using 1 x 1 detection kernels on feature maps of three different sizes at three places in the network, and a total of 255 kernels will be applied and convert the size of the output into 255 dimensions. These three layers are of different sizes and help to detect smaller objects in a scene. Each detection layer has a different output size as 13×13×255, 26×26×255, and 52×52×255, which hierarchically improves the detection performance.

The object detection model gives the location of the bounding boxes in an image. In an image I, suppose there are n bounding boxes. Each bounding box has (C_i, H_i, B_i) values: centre point, height, and width, respectively.

3.3.2 A Sequential Learning for Odometry Estimation

In the proposed model, the output of the 13×13×255 detection layer is given straight to the Conv-LSTM layers based network, which is a feature matrix $(f1, ..., fn)$. Based on the associated odometry value, the pose estimation

network conducts sequence learning of the relevant frames. Because LSTM units can handle sequence dependency, they are an excellent choice for learning odometry changes. Odometry is estimated using three values, two translations, and Yaw movements. The LSTM network is based on [25]; however, instead of utilizing conventional LSTM blocks, Conv-LSTM blocks were used. While processing the present input, it performs admirably. Conv-LSTM, which represents inputs, outputs, hidden states, and gates using a 3-D tensor, was employed. The spatial dimensions, which include rows and columns, are the final two dimensions.

3.4 Model Optimization and Loss Functions

To compute the conditional probability of the poses $P_t = (p_1,...,p_t)$ given for a sequence of monocular RGB images $I_t = (i_1,...,i_t)$ up to time t in the probabilistic perspective, and to find the optimal hyper parameters θ^*, the proposed model needs to optimize the following equation,

$$p(P_t \mid I_t) = p(p_1, ..., p_t \mid i_1, ..., i_t) \tag{1}$$

For training the model, we use the mean squared error (MSE) as a contents loss function. The MSE value between ground poses $(p_{xk}, p_{yk}, p_{\omega k})$ at time k and predicted poses $(\hat{p}_{xk}, \hat{p}_{yk}, \hat{p}_{\omega k})$ is minimized to learn the hyper parameters (θ) of the network.

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{t} \|\hat{p}_{xk} - p_{xk}\|_2^2 + \|\hat{p}_{yk} - p_{yk}\|_2^2 + \lambda \|\hat{p}_{\omega k} - p_{\omega k}\|_2^2 \tag{2}$$

where $\|.\|$ is a 2-norm, N is the number of samples in a sequence. λ is a constant, and it is used to maintain similarity between translation and orientation change. The orientation value ω is the Yaw angle, and it is used in the Euler representation. We did not use quaternion because it hinders operating with the deep learning networks.

3.5 Distance and Size Measurement of Objects

Object detection gives the corresponding bounding box with (C_i, H_i, B_i) values for all the images' objects. Now, the first task is to identify the objects located on the floor. In an image of dimension H and W, the ground object classification is done by a specific value of height pixel h_t. It should be noted that h_t is equivalent to the pixel that corresponds to camera height and is fixed for the whole process. All the objects with the bottom side of the bounding box below the h_t are considered as floor objects. As shown in Figure 3, the dotted horizontal line corresponds to the h_t. After confirming the presence of an object on the floor, it is sure that the pixels below the lower side of the bounding box of the object represent the ground or floor. By considering all ground pixels in a horizontal manner, the distance of the object is calculated.

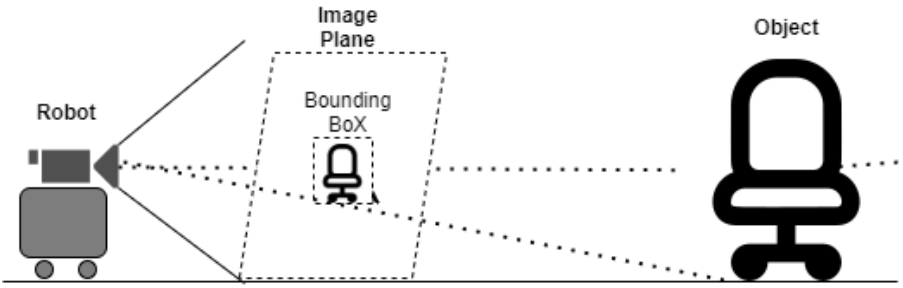

Fig. 3 Camera mounted on a Robot. Threshold pixel corresponds to the dotted line parallel to ground. The lower side of bounding box is considered for calculation. A robot with a camera attached to it. The dotted line parallel to the ground corresponds to the threshold pixel. For calculations, the lower side of the enclosing box is taken into account

The distance to the object is calculated as follows:

$$Z = h/\tan(\theta_h - \theta_o), \tag{3}$$

where Z is the distance perpendicular to the image plane, h is the height of the camera, θ_h is the angle of projection till h_t and θ_o is the angle of projection till the lower side of the bounding box.

For estimating the width and height of the object, Z is used as a reference, and it is calculated as follows:

$$(width, height) = Z * ((x_0, y_0) - (x_i, y_i))/f, \tag{4}$$

where (x_0, y_0) are the center points of the image plane, (x_i, y_i) are the center point of the bounding box along the width of the image plane, and f is the focal length of the camera.

3.6 2D Map for Floor Objects

We used a 2D map of the plane of the X and Z axes to represent the detected objects based on their locations in relation to the robot and for localizing the objects in the scene. The goal of employing a 2D map is to reduce the amount of computing power and memory required for its storage. In [49], point representation of space objects is used for localizing object points in semantic maps.

Three representations have been included, as shown in Figure 4: the input color image acquired by the camera, the matching computed distance from the robot, and a representation of the object on a 2D map. The distance on both axes is determined exactly, as evidenced by the 2D map. The displayed map is very light in operation and simple to utilize for other purposes, such as navigation. Any operation can be performed using simply the position coordinates when using the dotted representation.

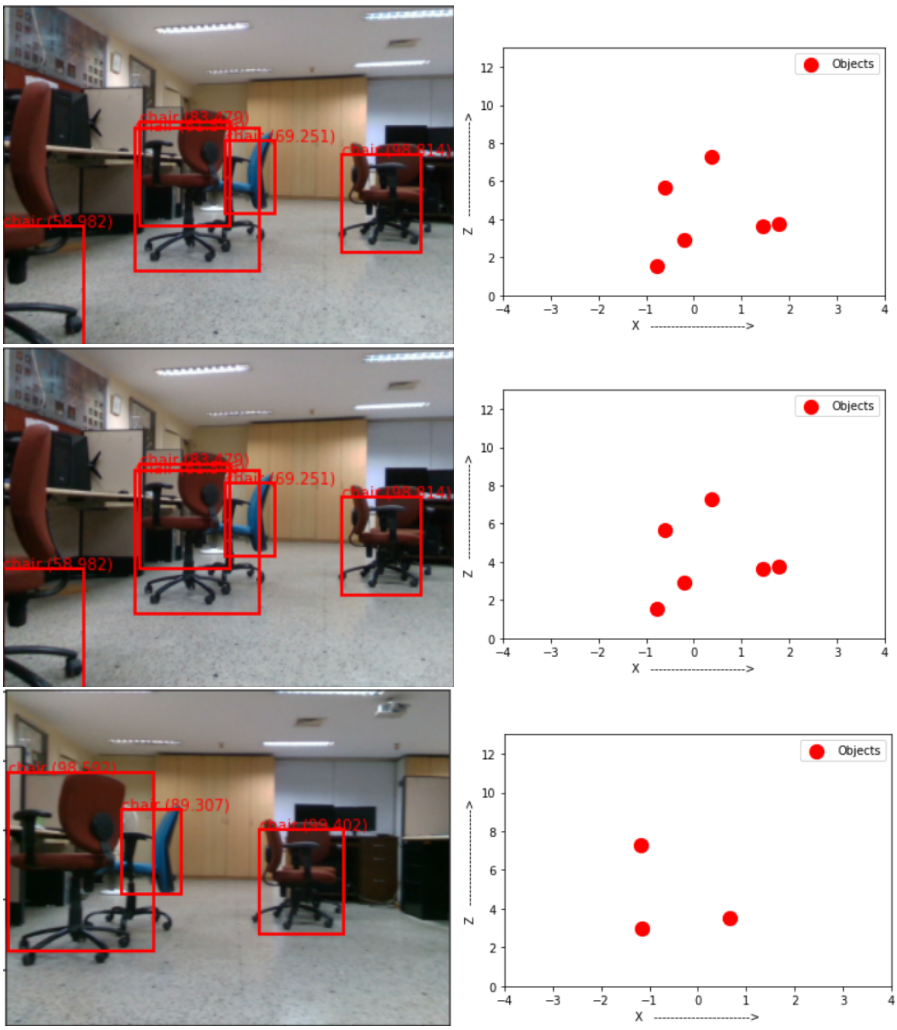

Fig. 4 Predicted distance of the objects using bounding box dimensions and a representation of object location on a 2D map for the floor location of the objects

3.7 Long Scale Map

A 2D map shows the location of objects with respect to the robot base. In the case of a robot with a moving base, dynamic frames will be captured with the camera, and a 2D map can be created corresponding to each frame. It is essential to integrate frame-wise operation and use a hierarchical clustering operation to connect the object points for the same entity in two different frames.

Suppose an object is represented as L_i in a map m_i for a frame i and the location of the same object in map m_{i+1} is L_{i+1} for the frame $i + 1$. The

relation between the two locations is then,

$$\vec{L_{i+1}} = M_{i-i+1} \cdot \vec{L_i}, \tag{5}$$

where M_{i-i+1} is the transformation matrix between the two maps that can be represented as:

$$M_{i-i+1} = \begin{pmatrix} \cos\theta_{change} & -\sin\theta_{change} & 0 & X_{change} \\ \sin\theta_{change} & \cos\theta_{change} & 0 & Z_{change} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6}$$

where, X_{change} and Z_{change} are the change in translation components and θ_{change} is the change in rotation component of odometry.
This relation gets violated in our experiment due to the presence of error in calculation of distance and the error in calculation of Z, which can be defined as follows:

$$Z \propto (-1/(\tan\theta_o)) \tag{7}$$

where θ_o is the angle of projection till the lower side of the bounding box.

It is obvious that object location in the map is a function of bounding box coordinates, which are not always exact due to the lack of convolution operators. Each time the location is determined, an error in position is introduced, which is eliminated using the clustering method. We have used agglomerative hierarchical clustering [43] to detect an object with various location by prediction on different frames and assign a single point on the map as the center of the cluster.

Agglomerative clustering is a bottom-up approach, which treats every data point as a cluster and merges them to make a new cluster. In the proposed method, the selection of points for merging is based on the Euclidean distance between points, and if a point with the same label lies in the threshold vicinity of the other point, then both points or clusters will be merged to make a new cluster. In our experiment, we have selected the threshold value of 1 meter, and it is decided empirically.

In Figure 5, three subfigures are shown in which (a) includes object locations on a global map by combining information from all frames, (b) includes clusters formed by using agglomerative clustering, and (c) shows the cluster centres for a unique object position. We have taken all the instances in a single frame, but in real scenarios, the clustering takes the locations of the objects after every frame. We use Euclidean distance to calculate the distance $d(A, B)$ between the same object points in two consecutive frames, and if the distance (d) is below a threshold, then both the points will be merged.

$$d(A, B) = (x_A - x_B)^2 + (y_A - y_B)^2 \tag{8}$$

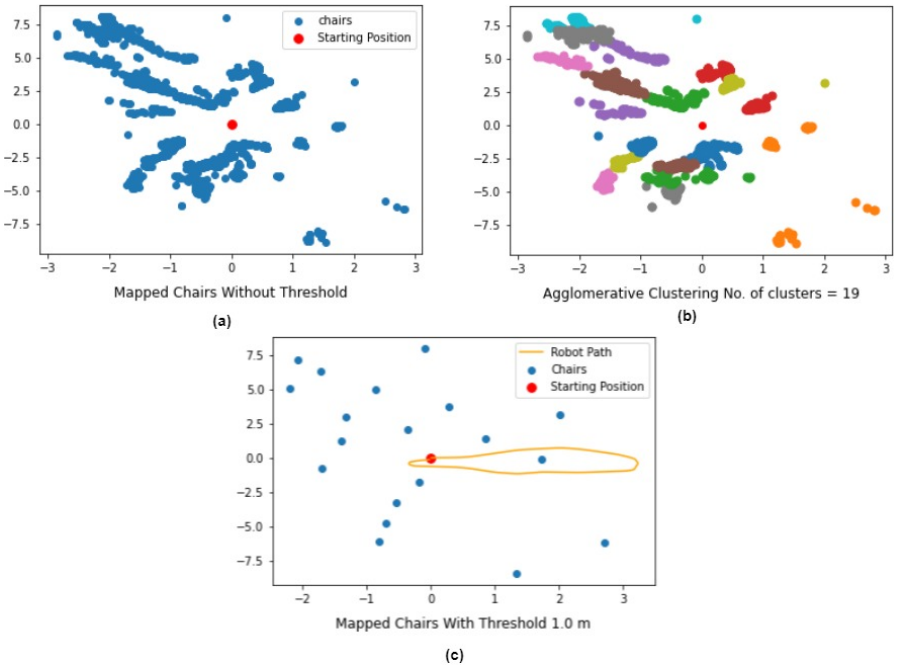

Fig. 5 In image (a), all generated points by the locations of the objects are shown, including the calculation from different locations. In image (b), a cluster wise formation is visible for each object. In image (c), all the cluster centres are shown. We have taken a threshold value of '1 meter' for making clusters

4 Experimental Results and Discussion

In this section, we describe the training process and discuss the results of the proposed model. There are also comparisons with other existing approaches and limitations.

4.1 Training the Proposed Model

The model is trained and tested on our private dataset. In our dataset, a total of 13000 sequence images are present, which include variations in the scenario. In total, 13000 sequence images, the first 5000 sequence frames are used to train the model, and the rest 8000 images are divided into four different robot trajectories. In Figure 7, the predicted trajectories using the trained model are shown, in which (a) shows prediction on train sequence, (b), (c), (d), and (e) shows prediction on test sequences.

The model was trained for 500 epochs, and it was trained on NVIDIA k40 GPU with 12 GB RAM. To reduce training time, the CNN part uses the pre-trained weights of the YOLO network trained with the COCO dataset [44]. The training was achieved for the pose change using corresponding odometry values. In order to meet YOLO v3 specifications, the image size is downsampled to 416×416. In selecting the model, we chose an AdaGrad optimizer, and the

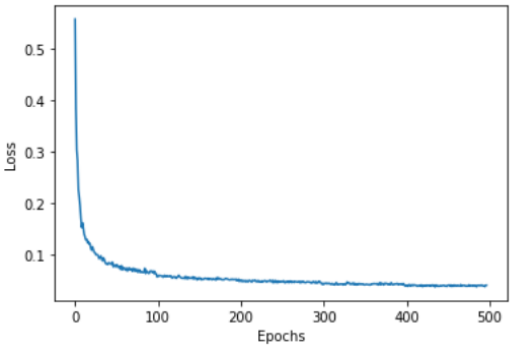

Fig. 6 The loss versus epoch graph to show the convergence of loss value while training on the own-private dataset. The model is trained on different sequences. The model is trained for 500 epochs

Table 2 Performance comparison for Object Detection model

Sequence	Total Chairs	Chairs detected	False Detection	Not Detected
1	124	116	2	10
2	388	356	9	41
3	402	358	11	55
4	206	192	8	22

learning rate was constantly decreasing from 0.005 to 0.001 after 100 iterations to prevent overfitting. Figure 6 shows the training loss graph of the model over 500 epochs.

4.2 Object Detection Results

The object detection is calculated in the context of importance to semantic mapping, and it is estimated for complete detection and false detection. In Table 2, the object detection capacity of the model is shown to detect chairs in different sequences of the dataset. It is more like the performance of YOLO v3, and the effect is just because of the scenario settings. We have counted the detection for 100 random frames from different sequences and manually annotated the visible objects in all the frames.

4.3 Visual Odometry Results

The proposed model is trained on a long sequence of 5000 frames and then tested on four different trajectories. The test scenarios are specially selected on the basis of variation in the trajectory. As shown in Figure 7, first trajectory is on the training data and rest all four trajectories are for test data, have different prediction accuracy. The reason for getting less accurate prediction seems the similarity between the test and train scenario. It is evident that the test cases with more similarity to the train scenario have better prediction

results and in case of more change of scenario, the results get deteriorated. In Table 3, averaged root mean squared error (RMSE) (ϵ) is used to calculate the preciseness in prediction for all five trajectories.

$$\epsilon = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x_i})^2}{N}} \tag{9}$$

where x_i, $\hat{x_i}$, and N represent the actual values, predicted values, and the number of points, respectively.

The model is used for predicting odometry by using sequence RGB images for five sequences as shown in Figure 7. In first plot, the prediction trajectory follows exactly near the ground truth and it is on training dataset, in second and third plot, the deflection from the ground truth is not much but in fourth and fifth plot,the predictions are too bad. A reason for this phenomenon is not having a proper refining of the dataset (due to irregular light conditions in the lab environment). To address this issue, we have tested this model on KITTI dataset.

4.3.1 Validation on an external dataset

The model is tested on the KITTI dataset for visual odometry. For the KITTI dataset, the same network is trained for 300 epochs. The input images are also resized to 416×416 before feeding to the network. The process is trained on sophisticated hardware because of the size of the KITTI dataset.

We compared the results of the proposed method with DeepVO [33] and VISO2 (monocular) [45]. The DeepVO is a well-known method with the LSTM network for learning visual odometry, and VISO2 is a sparse feature-based method for calculating the change in odometry. Table 4 shows the RMS error for different trajectories of the KITTI dataset. Our model is tested on the identical sequences for which we had the DeepVO and VISO2. The RMS error is calculated for the performance measurement of the model. Our model could not outperform the DeepVO model, but it gives satisfactory performance according to our application. The reason for this is the presence of a dedicated feature extractor in the DeepVO model, and in our case, we have used an object detection model.

4.3.2 Comparing the computational cost

In terms of computational cost, the proposed model has only 2,509,555 trainable parameters, which are significantly less than the DeepVO model and reduce the requirement of a sophisticated GPU unit for training. In the case of VISO2, our model outperforms the VISO2 in terms of mean squared error, which shows the effectiveness of the model and the relevance of using feature space for odometry learning in our model.

The proposed model is quite light and works well with loCoBot's computer. With odometry and object identification, it achieves a frame rate of 8-10

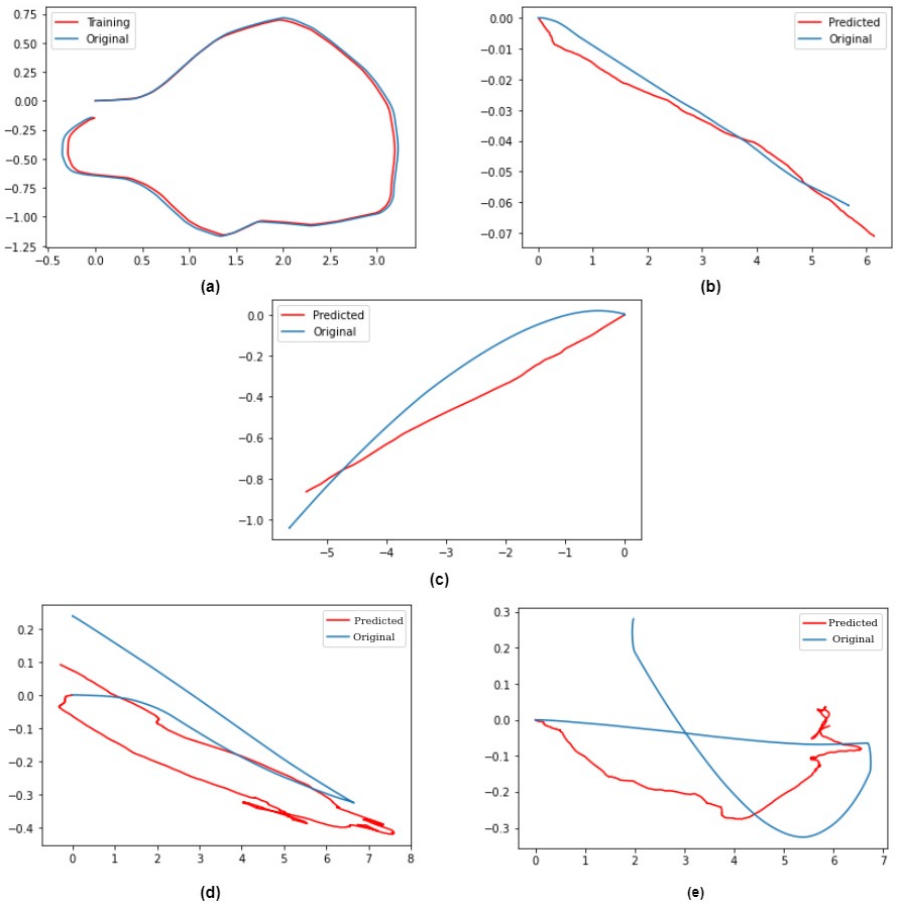

Fig. 7 The proposed model's visual odometry outcomes with five different sequences from the private dataset. To verify the model's robustness, all of the frames were taken from various motion types. (a) trajectory shows the test results on training sequence. (b) and (c) contains smooth trajectories.(d) contains sharp turn by the robot. (e) contains test set in varrying light condition

frames per second. DeepVO, on the other hand, performs poorly on CPUs and cannot be used for object detection. Due to the low intensity of light, VISO's performance is abysmal, and feature extraction and matching are quite poor.

4.4 Mapping of Objects

The mapping of the floor objects is being done based on the Manhattan world assumption. It states that all the sides of an indoor scenario are vertical, and the floor is always horizontal. We assume a small constraint in our work, considering objects below the camera height as floor objects. This assumption is not correct in all senses, but it relieves using time-consuming alternative techniques and expensive hardware for the mapping.

Table 3 RMSE with various sequences from the private dataset

Sequence	Accumulated Total loss
1	26.7502
2	6.9643
3	38.6718
4	48.1416
5	106.3202

Table 4 RMSE values with various sequences from the KITTI dataset and the number of trainable parameters of the proposed model, Deep VO [33] and VISO2 (mono) [45]

Sequence	Our Model	Deep VO [33]	VISO2 (mono) [45]
04	15.51	7.19	4.69
05	13.62	2.62	19.22
06	20.16	5.42	7.30
07	14.45	3.91	41.56
Average RMSE	15.93	4.78	18.19
Trainable Parameters (in millions)	2.5 M	230 M	NA

Table 5 Comparing the performance of different semantic mapping techniques in terms of hardware used, backbone network, sensors, and odometry model

Study	Hardware used	Backbone network	Sensors used	Odometry model
Meaningful Maps [10]	Tesla Titan X	SSD Camera	RGB-D	ORB-SLAM2
Panoptic Fusion [30]	Nvidia GTX1080	PSP-net Mask R-CNN	Scannet	Visual SLAM
Map using SLAM [31]	NA	Faster R-CNN	RGB-D Camera	CT-Map
SLAM-OR [32]	Tesla K80	Retina net YOLO v4	OID v4	ORB-SLAM
Our Model	Intel NUC with 8 GB Ram	YOLO v3	RGB Camera	Conv-LSTM

A comparison of the proposed model and other state-of-the-art odometry estimate approaches in indoor semantic mapping is shown in Table 5. Hardware, backbone network, sensors, odometry model, and map quality are all compared. RGB frames are used as input and output in the proposed model. It provides objects as well as odometry changes. This information is transformed into a 2D map via a different mapping model, which may locate previously visited objects. This map can also improve human-robot interaction and for high-level robotic tasks.

4.5 Limitations

We attempted many object detection architectures and integrated them with LSTMs in our studies, but YOLO proved to be the most effective. Additionally, solely utilizing LSTM made training more challenging and resulted in a higher parameter number. Furthermore, we made the fascinating finding of stacking ConvLSTM on all YOLOV3 outputs and predicting the results. The $13 \times 13 \times 512$ feature matrix can capture the small objects in the image quite well, and it works well in our scenario. The convergence of the loss finds this layer to be more compatible. We also attempted using other layers, however this increased the number of parameters and the results were likewise unsatisfactory. Changing the model's hyperparameters also proved to be highly useful. Changing the activation from regular ReLU to Leaky ReLU, for example, improves training time, and reducing the padding size for the Conv LSTM from (5,5) to (3,3) improves the proposed model's performance.

Although the proposed approach provides sufficient semantic mapping accuracy with low computational cost, it fails to predict the correct odometry in some cases. It appears to be an out-of-distribution example in which the scenario shifts and the light intensity changes. In some mapping scenarios, as such, two objects are in close proximity, causing clustering confusion and incorrect cluster creation.

5 Conclusion

This paper presented an efficient end-to-end deep learning-based semantic mapping technique based on a YOLOv3 object detector as a backbone and a convolutional long-short term recurrent neural network (Conv-LSTM) to model changes in camera pose. This technique eliminates the need for two separate pipelines for object localisation and semantics extraction. The proposed method varies from existing methods that employ complicated fusion techniques to match the data collected from diverse sources. The YOLO v3's backbone helps the model extract key features that aid in object detection and mapping. The proposed approach yields promising results on an own-private dataset collected in an indoor environment and the well-known KITTI benchmark.

The proposed model can be employed in a more detailed mapping application by combining depth information with odometry and object detection. It can help improve the robot's localization and improve object localization. A depth prediction network might be added to this model in the future to increase the quality of the semantic map and change the feature extraction layer by simultaneously employing the information for pose change and object detection.

6 Data Availability Statement

No Associated Data Available.

7 Conflict of Interest

The authors declare that they have no competing interests.

References

[1] Han, X, Li, S, Wang, X, Zhou, W (2021) Semantic Mapping for Mobile Robots in Indoor Scenes: A Survey. Information 12(2), 21, 4734. https://doi.org/10.3390/ s21144734

[2] Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp. 779-788. IEEE

[3] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: Single shot multibox detector. In: European conference on computer vision (ECCV), pp. 21-37. Springer, Cham

[4] Krul, S, Pantos, C, Frangulea, M and Valente, J, (2021) Visual SLAM for indoor livestock and farming using a small drone with a monocular camera: A feasibility study. Drones 5(2), p. 41

[5] Alsadik, B and Karam, S (2021) The simultaneous localization and mapping (SLAM)-An overview. Surveying and Geospatial Engineering Journal 2(01):01-12

[6] Ismail, H, Roy, R, Sheu, LJ, Chieng, WH, Tang, LC (2022) Exploration-Based SLAM (e-SLAM) for the Indoor Mobile Robot Using Lidar. Sensors 22(4), p.1689

[7] Pham TT, Reid I, Latif Y, Gould S (2015) Hierarchical higher-order regression forest fields: An application to 3d indoor scene labelling. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp. 2246-2254. IEEE

[8] Mozos OM, Triebel R, Jensfelt P, Rottmann A, Burgard W (2007) Supervised semantic labeling of places using information extracted from sensor data. Robotics and Autonomous Systems 55(5): 391-402

[9] Vineet V, Miksik O, Lidegaard M, Nießner M, Golodetz S, Prisacariu VA, Torr PH (2015, May) Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 75-82. IEEE

[10] Chen, Y, Zhang, J and Lou, Y, (2021) Topological and Semantic Map Generation for Mobile Robot Indoor Navigation. In: International conference on intelligent robotics and applications, pp. 337-347. Springer,

Cham

[11] Maolanon P, Sukvichai K, Chayopitak N, Takahashi A (2019) Indoor room identify and mapping with virtual based slam using furnitures and household objects relationship based on cnns. In: 10th International conference of information and communication technology for embedded systems (IC-ICTES), pp. 1-6. IEEE

[12] Narita G, Seno T, Ishikawa T, Kaji Y (2019) Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 4205-4212. IEEE

[13] Nistér D, Naroditsky O, Bergen J (2004) Visual odometry. In: Proceedings of the 2004 IEEE conference on computer vision and pattern recognition (CVPR), Vol 1, pp. 1-8. IEEE

[14] Davison AJ, Reid ID, Molton ND, Stasse O (2007) MonoSLAM: Real-time single camera SLAM. IEEE transactions on pattern analysis and machine intelligence, 29(6):1052-1067

[15] Davison AJ (2003) Real-time simultaneous localisation and mapping with a single camera. In: IEEE International Conference on Computer Vision, Vol. 3, pp. 1403-1403. IEEE

[16] Davison AJ, Reid ID, Molton ND, Stasse O (2007) MonoSLAM: Real-time single camera SLAM. IEEE transactions on pattern analysis and machine intelligence 29(6): 1052-1067

[17] Civera J, Davison AJ, Montiel JM (2008) Inverse depth parametrization for monocular SLAM. IEEE transactions on robotics 24(5):932-945

[18] Martinez-Cantin R, Castellanos JA (2005) Unscented SLAM for large-scale outdoor environments. In: 2005 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 3427-3432. IEEE

[19] Chekhlov D, Pupilli M, Mayol-Cuevas W, Calway A (2006) Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In: International symposium on visual computing, pp. 276-285. Springer

[20] Holmes S, Klein G, Murray DW (2008) A square root unscented Kalman filter for visual monoSLAM. In: 2008 IEEE International conference on robotics and automation (ICRA), pp. 3710-3716. IEEE

[21] Klein G, Murray D (2007) Parallel tracking and mapping for small AR workspaces. In: 6th IEEE and ACM international symposium on mixed and augmented reality, pp. 225-234. IEEE

[22] Klein G, Murray D (2008) Improving the agility of keyframe-based SLAM. In: European conference on computer vision (ECCV), pp. 802-815. Springer

[23] Mur-Artal R, Montiel JMM, Tardos JD (2015) ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics 31(5): 1147-1163

[24] Newcombe RA, Lovegrove SJ, Davison AJ (2011) DTAM: Dense tracking and mapping in real-time. In: 2011 International conference on computer vision (CVPR), pp. 2320-2327. IEEE

[25] Stadnik AV, Sazhin PS, Hnatic S (2020) Comparative performance analysis of neural network real-time object detections in different implementations. In: EPJ Web of Conferences, Vol. 226, p. 02020. EDP Sciences

[26] Coughlan J, Yuille AL (2000) The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference. In: Proceedings of the 13th International conference on neural information processing systems, pp. 809-815

[27] Redmon J, Farhadi A (2018) Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767

[28] Abdel-Nasser, M., Mahmoud, K., Lehtonen, M (2021) HIFA: Promising Heterogeneous Solar Irradiance Forecasting Approach Based on Kernel Mapping. IEEE Access 9:144906-144915

[29] Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision meets robotics: The kitti dataset. The International Journal of Robotics Research 32(11): 1231-1237

[30] Narita G, Seno T, Ishikawa T, Kaji Y (2019) Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In: 2019 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 4205-4212. IEEE

[31] Zeng Z, Zhou Y, Jenkins OC, Desingh K (2018) Semantic mapping with simultaneous object detection and localization. In: 2018 IEEE/RSJ IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 911-918. IEEE

[32] Mazurek P, Hachaj T (2021) SLAM-OR: Simultaneous Localization, Mapping and Object Recognition Using Video Sensors Data in Open Environments from the Sparse Points Cloud. Sensors 21(14):4734

[33] Wang S, Clark R, Wen H, Trigoni N (2017) Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In:

2017 IEEE international conference on robotics and automation (ICRA), pp. 2043-2050. IEEE

[34] Kundu A, Li Y, Dellaert F, Li F, Rehg JM (2014) Joint semantic segmentation and 3d reconstruction from monocular video. In: European conference on computer vision (ECCV), pp. 703-718. Springer, Cham.

[35] Abdel-Nasser, M, and Mahmoud K. (2019) Accurate photovoltaic power forecasting models using deep LSTM-RNN. Neural Computing and Applications 31(7):2727-2740

[36] Lalapura, VS, Amudha, J Satheesh, HS (2021) Recurrent neural networks for edge intelligence: A survey. ACM Computing Surveys (CSUR) 54(4): 1-38

[37] Alzaidy R, Caragea C, Giles CL (2019) Bi-LSTM-CRF sequence labeling for keyphrase extraction from scholarly documents. In: The world wide web conference, pp. 2551-2557

[38] Jiao J, Jiao J, Mo Y, Liu W, Deng Z (2019) MagicVO: An End-to-End hybrid CNN and bi-LSTM method for monocular visual odometry. IEEE Access 7:94118-94127

[39] Kerl C, Sturm J, Cremers D (2013) Dense visual SLAM for RGB-D cameras. In: 2013 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 2100-2106. IEEE

[40] Taketomi T, Uchiyama H, Ikeda S (2017) Visual SLAM algorithms: A survey from 2010 to 2016. IPSJ Transactions on Computer Vision and Applications 9(1):1-11

[41] Keselman L, Iselin Woodfill J, Grunnet-Jepsen A, Bhowmik A (2017) Intel realsense stereoscopic depth cameras. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 1-10

[42] Pyrobot (accessed date: 21 March 2022). https://pyrobot.org

[43] Day WH, Edelsbrunner H (1984) Efficient algorithms for agglomerative hierarchical clustering methods. Journal of classification 1(1):7-24

[44] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Zitnick CL (2014, September) Microsoft coco: Common objects in context. In: European conference on computer vision (ECCV), pp. 740-755. Springer, Cham

[45] Geiger A, Ziegler J, Stiller C (2011) Stereoscan: Dense 3d reconstruction in real-time. In: 2011 IEEE intelligent vehicles symposium (IV), pp. 963-968. IEEE

[46] Liu Y, Wang H, Wang J, Wang X (2021) Unsupervised monocular visual odometry based on confidence evaluation. IEEE Transactions on Intelligent Transportation Systems. Early access, 1-10, doi:10.1109/TITS.2021.3053412

[47] Pandey T, Pena D, Byrne J, Moloney D (2021) Leveraging deep learning for visual odometry using optical flow. Sensors 21(4):1313. https://doi.org/10.3390/s21041313

[48] Ban X, Wang H, Chen T, Wang Y and Xiao Y (2021) Monocular Visual Odometry Based on Depth and Optical Flow Using Deep Learning.IEEE Transactions on Instrumentation and Measurement 70:1-19, Art no. 2501619. doi: 10.1109/TIM.2020.3024011

[49] Han X, Li S, Wang X, Zhou W (2021) Semantic Mapping for Mobile Robots in Indoor Scenes: A Survey. Information 12(2):92. https://doi.org/10.3390/info12020092

[50] Liu Y, Sun P, Wergeles N, Shang Y (2021) A survey and performance evaluation of deep learning methods for small object detection. Expert Systems with Applications 172: 114602. https://doi.org/10.1016/j.eswa.2021.114602