Q-Loc: Visual Cue-Based Ground Vehicle Localization Using Long Short-Term Memory

Cole Malinchock, Jimin Yu, Pratik Thapa, Dhruva Ungrupulithaya, Man-Ki Yoon
North Carolina State University

***Abstract*— Mobile autonomous systems are increasingly being deployed in controlled environments worldwide, with large fleets of ground robots performing tasks such as delivery and surveillance. These systems require reliable localization to navigate through such environments. While the Global Positioning System (GPS) is commonly implemented in these systems, urban environments can introduce inaccuracies due to signal blockages caused by large buildings and structures, or even complete signal loss. This paper proposes a rapid and cost-effective localization method using a sensor ubiquitous in autonomous systems: cameras. We introduce a system that uses vision-based machine learning techniques to detect common landmarks in camera streams and subsequently predict location. The system employs advanced object detection models for landmark identification and recurrent neural networks for vehicle localization based on the detected landmarks. We prototype these techniques on a small-scale autonomous vehicle platform to demonstrate the system's capabilities and evaluate its accuracy and execution efficiency in real-world scenarios.**

I. Introduction

The incorporation of autonomous systems into various domains including last-mile delivery [1], food delivery [2], surveillance [3], and military applications [4] has seen a rapid advancement in recent years. The successful operation of these systems hinges on reliable *localization* systems for tracking the position of the robot in its immediate environment and ascertaining its global coordinates. Accurate localization is crucial for the effective execution of location-based tasks such as path planning, site monitoring, and other key functions that enable autonomous operation.

For autonomous robots, sensors such as Global Positioning Systems (GPS) are often key components of their localization strategy. It provides a global context for the system's position, which is essential for navigation in outdoor areas. Nonetheless, the precision of GPS is substantially affected by environmental obstructions like buildings, bridges, and trees, which induce signal reflections and culminate in either erroneous or absent GPS measurements [5].

One potential solution involves the integration of advanced sensors, such as Real-Time Kinematics (RTK) units, which can achieve enhanced precision. Berntorp [6] proposes to employ wheel speed sensors, Inertial Measurement Units (IMUs), and GPS to determine location estimates. An alternative strategy involves the deployment of perception sensors, such as RGB cameras and LiDARs, to ascertain relative position [7]. The relative position derived from these perception-based localization methods, when combined with position and motion sensors such as GPS and IMU, can augment the accuracy of the measured global position [8].

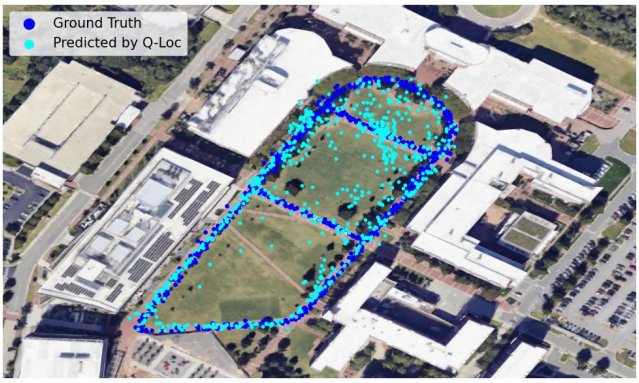

Fig. 1. A satellite image of the area of operation of our ground autonomous robot. The dark blue points represent the GPS coordinates when measured using a RTK module. The light blue points represent the coordinates predicted by Q-Loc.

However, adding these sensors may significantly increase both the initial capital expenditure and the operational expenses of these autonomous robots. A typical RTK and GPS module with centimeter-level precision or a 3D LiDAR for LiDAR odometry can cost thousands of dollars per module [9], [10], [11]. As commercial entities attempt to scale their operations, they may face considerable financial challenges due to the compounding of expenses required to acquire these sensors, potentially making the technology economically unfeasible. Therefore, there is a clear need for a cost-effective localization algorithm that still maintains the desired accuracy. The key research challenge can be articulated as follows: *how do we develop a localization mechanism with the least amount of sensors?*

Cameras are inexpensive and versatile sensors that are an irreplaceable component in autonomous systems today that provide critical information about the environment. The camera data is utilized for various tasks such as object detection and tracking, lane recognition, remote teleoperation, or even simple data collection and logging. This serves as an inspiration for this paper: *localization using only cameras*. This *single-input* approach to localization can help eliminate the reliance on other, often expensive sensors, drastically reducing the capital expenditure required to develop autonomous robots operating in constrained geographic locations.

Visual cues play a crucial role in spatial localization for humans and other animals, providing a rudimentary yet highly accurate mechanism for navigating and orienting within an environment. Static objects, such as buildings, act as landmarks that aid in position estimation, which is a principle that is already utilized in localization algorithms.

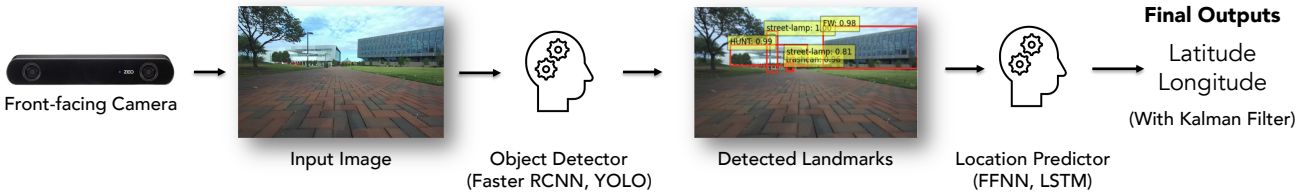

Fig. 2. Overall execution flow of Q-LOC. It learns to detect visual cues and features from the environment, such as buildings or other recognizable structures, and predicts the vehicle's latitude and longitude based on the detected landmarks.

Betke et al. [12] propose using indoor landmarks like doors, walls, and picture frames to determine a robot's position. Lu et al. [13] propose a method to extract the inherent geometric features of man-made landmarks such as buildings and signs to account for positional errors. Using visual cues for localization can be effectively extended to autonomous ground vehicles by leveraging their existing cameras.

In this paper, we present Q-LOC, a solely monocular camera-based approach to localization. Our framework uses landmarks in an input image to predict the position of the vehicle. Q-LOC identifies landmarks in the vicinity of the vehicle by utilizing real-time object detection models, specifically Faster Regional Convolutional Neural Network (FASTER R-CNN) [14]. The presence and relative position of these landmarks in the scene are then fed to a Long Short-Term Memory (LSTM) based model which predicts the current *latitude* and *longitude*, taking into account the vehicle's previous positions and the detected landmarks.

To achieve this, we deploy a custom-built data acquisition robot that captures camera images with the simultaneous, high-precision GPS coordinates using an RTK module. It is important to note that the GPS is only required for initial data collection, i.e., when deploying the system in a brand new environment, and thus can be treated as a *one-time cost* for data acquisition. Once sufficient data is collected for training, we no longer require expensive GPS and RTK modules, making Q-LOC an easy and cost-effective approach to deploy large fleets of autonomous ground robots operating within the same geographical area.

Q-LOC can predict the latitude and longitude of the vehicle with an accuracy of 16.57 meters on average using purely visual cues (see the light blue points in Figure 1), while other existing methods such as [15] achieve an accuracy of 95 meters on average when applied to our dataset.

In this paper, we make the following contributions:

- We present a *purely vision-based, single-sensor* approach to global localization that leverages object detection models to identify landmarks, which are then used to predict the vehicle's location.
- We analyze the Q-LOC design parameters and their impacts on the prediction accuracy and execution efficiency of the localization.
- We show the practicality of Q-LOC by applying it to a prototype ground robot vehicle and evaluating its performance in real-world scenarios.

II. RELATED WORK

Several works explore localization using perception-based algorithms, some of which are summarized as follows:

a) 3D Feature Mapping: These systems employ feature mapping techniques, achieving high levels of accuracy and robustness by discerning distinguishing features inherent in images that aid in localization [16], [17], [18]. Some works utilize additional sensor inputs such as coarse GPS location, camera height, and gravity direction during inference [19], [20]. However, systems utilizing feature mapping generally require 3D maps with large memory footprints, as they store dense 3D point clouds with high-dimensional visual descriptors. Some works have attempted to compress these maps, although such methods result in a significant loss in accuracy [21], [22].

b) 2D Image Pose Estimation: Sarlin et al. [23] attempt to tackle the challenges associated with 3D maps by proposing a method using 2D maps, which are 10^4 times smaller in size than 3D maps. Nonetheless, the necessity for birds-eye-view (BEV) transformations, feature matching with maps, and the storage of maps, even in their 2D form, results in computational and memory demands that are suboptimal for real-time inference in resource-constrained embedded platforms of autonomous systems. This also assumes that reliable, detailed maps exist for a certain area. Significant progress has been achieved in leveraging Structure-from-Motion (SfM) algorithms on sequentially captured 2D images processed in real-time through neural networks. This approach enables the use of more compact models while maintaining accuracy comparable to larger 3D maps [24]. Although this method is able to compete with the overall accuracy of state-of-the-art methods, they are all relative motion-based and lack global positioning [25], [26], [27]. The inability to provide global positioning presents a significant challenge in environments where accurate GPS data is rarely available.

c) SLAM Positioning: Cao et al. [28] evaluate a method of implementing LiDAR's BEV image-based SLAM for outdoor relative localization, which utilizes the LiDAR BEV images to decrease the computational strain. Certain approaches also use vision-based SLAM algorithms for relative motion estimation [29][30]. Nevertheless, these methodologies necessitate high-cost sensors, as well as significant memory capacity, to store the maps, particularly when applied to extensive geographic regions. Similar to the 2D Image Pose Estimation approach, they are only relative to their own motion and lack the ability to provide global position.

Global localization can be attained through the utilization of radio frequency (RF) sensors as well [31]. However, the precision of an RF-based methodology is contingent upon factors such as the orientation of antennas and the density of beacons [32]. Moreover, RF-based techniques are susceptible to signal interference and loss, similar to GPS-based methods [31], [32]. As higher sampling frequencies are required to have sufficient spatial resolution, these solutions also require significantly more resources to deploy in a practical environment [33].

III. Q-Loc: Ground Vehicle Localization using Visual Cues

Q-Loc is based on the hypothesis that given a sequence of images, we can predict the current location with sufficient precision if the unique features of the encountered scene are correctly identified. Additionally, we hypothesize that the latitude and longitude can be inferred given an appropriate set of features.

Q-Loc has two steps as shown in Figure 2:

1) **Step 1 - Landmark Detection:** The camera stream is first fed into a convolutional neural network (CNN) based object detection model which detects the landmarks visible in the image. The object detection model is trained to identify static, commonplace objects such as signs and street lamps, as well as distinctive architectural structures.
2) **Step 2 - Location Prediction:** The detected landmarks, along with their relative positions and scales in the scene (visual cues), are then fed into a deep learning model that predicts the vehicle's *latitude* and *longitude*. These outputs are then processed through a Kalman filter to produce a more accurate location estimate for the vehicle.

A. Target Systems and Environments

Q-Loc demonstrates optimal performance when deployed in distinct small geographic settings, as we will see in Sections IV and V. Autonomous robots, employed for purposes such as delivery, surveillance, and maintenance, predominantly function within a *fixed geographic location*, thus making our methodology pertinent [15], [8]. This approach has multiple benefits: a small geographic setting would require less data for training the models in Q-Loc, consequently reducing the data-acquisition and training costs drastically. A smaller environment also allows users to leverage unique features, like buildings and street signs, to develop more effective localization models, which generic approaches may not be able to do. The resulting algorithm is computationally less expensive than generic approaches as well. In contrast, generic approaches like 3D feature mapping rely on collection of high quality data to develop their resource intensive 3D maps on predefined geographical locations [26], [27]. Scaling these 3D models would result in an increase in the amount of resources required for both data acquisition and deployment. Lee et al. [34] propose a generic visual localization approach that requires a server-side visual localization algorithm in addition to a client-side visual odometry algorithm, significantly increasing the resources required for large-scale deployment. Q-Loc requires a relatively inexpensive data acquisition phase for the target geography to develop and deploy a real-time visual localization algorithm across a large fleet of autonomous ground robots. Only a single robot equipped with a camera and GPS + RTK module is needed to acquire training data.

B. Step 1 - Landmark Detection

Due to the computational constraints of robotic platforms, selecting object detection models requires balancing accuracy and inference latencies. State-of-the-art object detection algorithms like Detection Transformers (DETR) perform exceptionally well at their tasks but come at the cost of high computation requirements, making them unsuitable for real-time applications [35], [36]. On the other hand, CNNs have a lower computational cost compared to transformers and are better optimized for resource-constrained platforms [37], [38]. We hence evaluate two popular CNN-based object detection models: Faster R-CNN [14] and YOLOv8n [39], each offering distinct trade-offs between accuracy and efficiency.

1) Faster R-CNN for Landmark Detection: Faster R-CNN typically comprises two modules: (i) a regional proposal network (RPN) and (ii) a Fast R-CNN object detector. The structure of the latter contains several convolutional layers and max pooling layers, a region of interest (RoI) pooling layer, and a sequence of fully connected layers. The RPN is a deep CNN used to generate region proposals. It processes the image using the same convolutional layers used in the Fast R-CNN detection network, so it does not take extra time to produce the proposals. The RPN of Faster R-CNN first generates high-quality region proposals for a given image, following which a fixed-length feature vector is extracted from each region using the RoI pooling layer. The extracted feature vectors are then classified using Fast R-CNN. Finally, the class scores of the detected objects, in addition to their bounding boxes, are returned. In this paper, we use PyTorch's Faster R-CNN ResNet50 FPN V2 [40] model. It uses a ResNet50 as a backbone while utilizing the object detection modules of Faster R-CNN. This model utilizes Feature Pyramid Network (FPN) V2, which improves multi-scale feature representation. This is especially useful for detecting objects of different sizes and scales, which is crucial for detecting landmarks in outdoor environments.

2) YOLO for Landmark Detection: Although Faster R-CNN is highly accurate, being a two-stage detector still results in higher inference times. YOLO [41] treats object detection as a regression problem, as opposed to region proposal like in Faster R-CNN. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. Since object detection is framed as a regression problem, there is no need for a complex model pipeline, resulting in rapid detection and

classification. Additionally, YOLO processes the entire image during both training and testing, allowing it to implicitly capture contextual information about object classes and their appearances. Region-proposal-based techniques like FAST or FASTER R-CNN, on the other hand, can mistake background patches as objects since they do not see the larger context. In fact, YOLO makes half the number of background errors compared to FAST R-CNN [41]. Lastly, YOLO learns generalizable representations of objects, meaning it is less likely to break down when applied to new domains or unexpected inputs. Despite its benefits, YOLO has trade-offs in accuracy compared to other state-of-the-art detection systems. Although it excels in detection speed, it struggles to precisely localize some objects, especially small ones [41]. The specific YOLO model used in this paper is YOLOV8N [42], [43], [44]. The YOLOV8 model was built upon the YOLOV5 model, introducing a new network structure and incorporating the strengths of previous YOLO models and other state-of-the-art design concepts in target detection algorithms [45].

C. Step 2 - Location Prediction using Long Short-Term Memory (LSTM)

LSTMs are designed to take advantage of temporal sequences to make predictions [46]. This makes them useful for localization as it can learn the *temporal dependencies* in sequential input data to predict the target variable, which in this case is the vehicle's location. LSTM has been shown to be effective in predicting robot locations (particularly in indoor environments) [47], [48], [49]. In Q-LOC, the LSTM network learns the relationships between the relative positions and scales of landmarks in an image (as bounding boxes) to predict the vehicle's location at that moment. This type of network accounts for the fact that, given the vehicle's previous location, its geolocational status must remain within a reasonable bound. This provides the model with additional context beyond the detected landmarks alone to aid in the prediction accuracy.

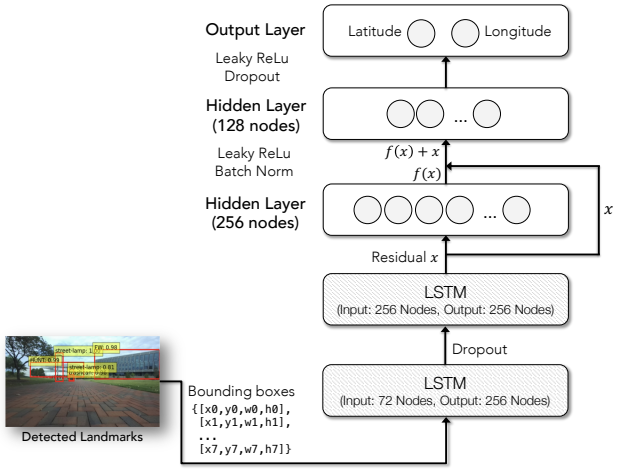

Fig. 3. LSTM-based location prediction using landmarks detected from a scene.

Model Architecture. The network structure of location prediction using LSTM is shown in Figure 3. The implemented model consists of 64 input neurons representing the detected landmarks. Each detected object is represented as a vector that contains four elements for the position and size: the coordinates relative to the image for the top-left point of the bounding box, then the width and height of the bounding box. The unit of position coordinates is based on the number of pixels in the image. For example, a bounding box that has a vector of $< 40, 30, 35, 24 >$ has its top-left corner at pixel number 40 horizontally and 30 vertically, and extends 35 pixels to the right, and 24 pixels down. In our experiments, we have eight landmark classes, which will be described later in Section IV. Because the input layer size is fixed, we limit the number of maximum detection instances of each class (e.g., two instances per class). For example, if there are three LAMP objects detected in an image, only the two with the highest classification scores are used in the input. This approach allows us to work with the most relevant features in the image. Since we have eight landmark classes, the number of input neurons can be calculated to be 4 x 8 x 2 = 64 input neurons. The LSTM outputs 128 neurons per time step and is bidirectional, hence outputting a total of 256 neurons. Using a bidirectional LSTM allows the network to learn both forward and backward dependencies in a sequence, which is useful for capturing the temporal relationships across a sequence of scene images. We stack two LSTM layers on top of each other, with the output of the first layer becoming the input of the second layer to compute the final results. This allows the network to learn more complex temporal dependencies in the sequence of scenes. A dropout is applied between the two LSTM layers to prevent overfitting.

The LSTM layers are followed by multiple fully connected layers. After the first fully connected layer, a skip connection is introduced, where the original output of the LSTM is added back to the output of this layer. This helps retain the raw sequential and spatial information from the LSTM and prevents the loss of critical features as they pass through the fully connected layers. It also helps address the vanishing gradient problem during training by providing a direct path for gradients to flow from later layers back to earlier ones [50]. This ensures that earlier recurrent layers, such as the LSTM, continue to learn effectively. Batch normalization and dropout are also applied after the first layer to ensure that the model generalizes well and avoids overfitting. Additionally, a Leaky ReLU activation function is used after each fully connected layer. This is preferred over the traditional ReLU, which can suffer from the "dying ReLU problem" – where neurons may never activate when inputs are negative, leading to ineffective weight updates [51].

Kalman Filtering. The location prediction from the LSTM model is then passed through a Kalman filter to produce a more accurate estimate of the vehicle's location. We define the state of the system as the vehicle's position and velocity:

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T,$$

where x and y represent the current position of the system in Cartesian coordinates in meters after the latitude and longitude are transformed using Equation (1) in Section V and setting the origin at a specified point at the center of the target environment. $\dot{x}$ and $\dot{y}$ represent the velocity of the vehicle in the x and y directions in meters per second. We use (i) a simple linear motion model, i.e., $x_{k+1} = x_k + \dot{x}_k \Delta t$ and $y_{k+1} = y_k + \dot{y}_k \Delta t$ and (ii) the independent noise model for the process noise covariance, and (iii) assume that the measurement noise for the latitude and longitude are independent.

IV. Experiment

A. Data Collection

The data collection process was conducted using a custom-built mobile vehicle that was driven on a university campus along a pathway that is roughly 750 meters in total length, as shown in Figure 4. To ensure a diverse dataset, the vehicle was driven along different possible routes on the pathway with varying lighting conditions.

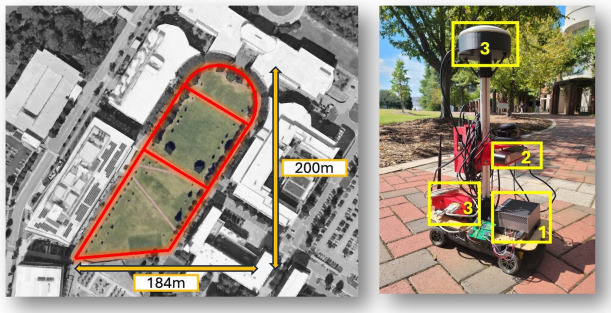

Fig. 4. Target environment, data collection path, and data acquisition vehicle.

The mobile vehicle was built off of a Traxxas 1/10 Scale 4-Tec 2.0 Brushless AWD Chassis and equipped with (1) an NVIDIA Jetson Orin Developer Kit [52], (2) a Zed2i Stereo Camera [53], and (3) a Swift Navigation RTK system. By using an RTK module, we compensate for the errors caused by using GPS sensors alone, resulting in highly precise measurements [**?**]. The vehicle and all components mounted on it for the data collection process are shown in Figure 4.

Every 2 seconds, the robot records the timestamp, RTK latitude, RTK longitude, and the 1280x720 image captured at that moment by the left-facing lens of the Zed2i camera. To collect the data, we implemented Robot Operating System 2 (ROS2) nodes that simultaneously record the data while the vehicle is driven around the target environment. Data was collected for approximately 120 minutes, yielding 3,458 data points along the route, which equates to an average of 4.6 data points per meter traveled.

The data collection process was conducted on two separate occasions: one dataset was used for training the YOLO and Faster R-CNN models, while the second dataset was used for training the LSTM and FFNN models. Out of the 3,458 total data points collected, 2,766 (80%) were allocated for training, and 692 (20%) were used for testing.

Fig. 5. Landmarks classes used in the experiments.

All images in the collected dataset were manually labeled with bounding boxes of landmarks in the images. Figure 5 shows the eight landmark classes. This labeling process was separated into two stages. During the first stage, the images were annotated with five landmark classes ('building', 'security-station', 'sign', 'street-lamp', and 'trashcan') using Roboflow [54], which is an online low-code tool used to build and deploy task specific computer vision models. The second stage involved further categorizing the landmarks labeled 'building' into the names of specific buildings around the target environment. To label specific building names, we developed an automated process using the OpenStreetMap API [55]. This process utilizes the current position and orientation to retrieve the name of the building within the bounding box based on the actual map information.

B. Model Training

The labeled images collected during the data acquisition phase were used to train the object detection models in Roboflow [54]. The ground truth bounding boxes and its corresponding classes were provided for training the models. The setup for training the LSTM model included training for 300 epochs with a batch size of 16. The learning rate was initially set to $1 * 10^{-3}$ and decreased by a factor of 0.1 every 100 epochs. Hyperparameter optimization was done by manual trial-and-error. The loss function used was Mean Squared Error (MSE), which is ideal for handling regression tasks.

Since the system was deployed in a small geographic area, the coordinate data remains static up to the tenth of a degree. This constricts the accuracy of the model to an area taken up by 0.1 degrees latitudinally and 0.1 degrees longitudinally, which corresponds to approximately 11.10 kilometers in the latitudinal direction and 9.02 kilometers in the longitudinal direction at a latitude of 38 degrees North [56]. Such narrowly distributed data can introduce challenges during training, including the vanishing gradient problem and slow weight updates. To address this, a modulo operation with scaling was applied to the coordinate data to map it to a wider range, as follows:

$$l' = (l \mod 0.1) \times 1000,$$

where l represents the original coordinate value (either

latitude or longitude). This results in retaining only the digits starting from the second decimal place onward, which are then multiplied by 1000 to shift the decimal point back to its original position. For example, the coordinate value 35.7658921 would be transformed to 65.8921. This operation was applied to the latitude and longitude data before training the models and reversed after they were predicted.

Feedforward Neural Network (FFNN). In addition to using LSTM for localization, the Feedforward Neural Network (FFNN) proposed by Nilwong et al. [15] is also implemented and tested as a baseline model. The FFNN used in this paper consists of 64 input neurons and five total fully connected layers: 256, 128, 64, 32 hidden neurons in the first, second, third, and fourth hidden layers, respectively, and two output neurons. The output of the first fully connected layer is batch-normalized, and a Leaky ReLu activation function is applied. A dropout with a probability of 0.3 is also applied following this layer. This model shares the same hyperparameters (batch size, learning rate, number of epochs) as the LSTM model.

V. Evaluation Results

Landmark detection and localization models were trained on a workstation with an NVIDIA RTX 4090 GPU with 24 GB of memory and 16,384 CUDA cores [57], and a 24-core Intel Core i9-13900f CPU. For real-time inference, the NVIDIA Jetson Orin Developer Kit [52] was used (see Figure 4). The Jetson Orin contains NVIDIA's Ampere architecture GPU with 2,048 CUDA cores and 64 tensor cores, and a 12-core ARM Cortex-A78AE v8.2 64-bit CPU.

Note that we consider two separate models for landmark detection in this paper – Faster R-CNN and YOLO – and two separate models for location prediction – LSTM and FFNN. Hence, we have four different combinations of models to evaluate: 1) Faster R-CNN + LSTM, 2) Faster R-CNN + FFNN, 3) YOLO + LSTM, and 4) YOLO + FFNN. Of these, the LSTM variants (i.e., 1 and 3) are the primary focus of this paper, while the FFNN variants are used as baselines for comparison.

To evaluate our method, we used a representative subset of the dataset described in Section IV, selected to capture diverse conditions such as lighting, perspective, and landmark density. The images were processed by the object detection model to extract bounding box coordinates and class labels, which were then passed to the latitude/longitude predictor to generate global coordinates.

We compare the execution efficiency and accuracy of the Faster R-CNN and YOLO models for landmark detection. Table I shows the mean absolute precision (mAP) along with the mean and standard deviation (SD) of the inference times when the landmark detection models are executed on the Jetson Orin. While YOLOv8n has lower mAP than Faster R-CNN, it is nearly 4.5 times faster than Faster R-CNN. At 30 frames per second, the YOLOv8n model can adequately perform landmark detection for each frame.

TABLE I
Faster R-CNN vs. YOLOv8n Performances

	mAP	Mean Inference Time (ms)	SD (ms)
Faster R-CNN	0.88427	109.50	5.25
YOLO	0.76406	24.45	1.98

Localization results were evaluated by calculating the mean absolute error (MAE) between the predicted values and the ground truth (collected using the RTK module), as well as the SD of these values. The errors were calculated by converting the distance between two latitude and longitude points to *meters* using the Haversine formula, as follows:

$$d = R \cdot 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right),$$

where

$$a = \sin^2\left(\frac{\phi - \tilde{\phi}}{2}\right) + \cos(\tilde{\phi}) \cdot \cos(\phi) \cdot \sin^2\left(\frac{\lambda - \tilde{\lambda}}{2}\right), \quad (1)$$

where d represents the spherical distance between an predicted location $(\tilde{\phi}, \tilde{\lambda})$ and the reference location (ϕ, λ) on Earth. R is the radius of the Earth in meters, 6,378,137 m.

TABLE II
Localization Errors

Model		YOLO	Faster R-CNN
FFNN	**MAE**	**49.53**	**33.73**
	Min	2.15	1.43
	Max	476.35	139.21
	SD	**40.43**	**21.19**
LSTM	**MAE**	**25.43**	**16.57**
	Min	0.03	0.11
	Max	162.96	102.49
	SD	**26.24**	**18.79**

Table II presents the average localization errors for each of the four combinations. The minimum, maximum errors, MAE, and SD were calculated from the absolute errors between the ground truth and predicted locations. The Faster R-CNN + LSTM method achieved the lowest MAE and SD of 16.57 meters and 18.79 meters, respectively. Meanwhile, YOLO + LSTM had a higher MAE of 25.43 meters but demonstrated the best minimum MAE of 0.03 meters. In contrast, both FFNN methods have lower accuracy across all metrics, with Faster R-CNN + FFNN yielding an MAE of 33.73 meters and YOLO + FFNN an MAE of 49.53 meters.

The results clearly show that the LSTM approach outperforms the FFNN for localization. Figure 6 highlights the trade-offs between speed and accuracy for each method. Although Faster R-CNN + LSTM yields the most accurate predictions, YOLO + LSTM offers significantly faster inference times, as highlighted in Table I. With an inference time of 0.11 seconds, the Faster R-CNN + LSTM method remains more than adequate for most applications, surpassing typical GPS devices that update at a rate of 1 Hz [58]. In scenarios requiring faster update rates or lower computational resources, the YOLO + LSTM method, though less accurate, could serve as a more efficient solution.

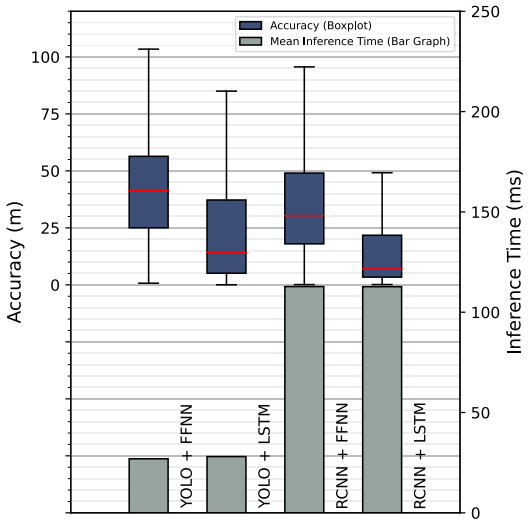

Fig. 6. Inference time and accuracy of different combinations.

TABLE III
ACCURACY COMPARISON BETWEEN OUR MODELS AND BASELINE MODEL.

Model	MAE (m)	SD (m)
Baseline [15]	95.44	32.63
Faster R-CNN + FFNN	33.73	21.19
Faster R-CNN + LSTM	16.57	18.79
Faster R-CNN + LSTM (Unobstructed)	10.32	21.10

Table III highlights the accuracy comparison between our FASTER R-CNN-based models and a baseline model that uses a single hidden layer with 48 neurons [15]. Since the source code was unavailable, we replicated the architecture and experimented with a learning rate of 8×10^{-3} using the Adam optimizer. Following the training process and model architecture outlined in [15], the baseline achieved an MAE of 95.44 meters. However, after introducing additional hidden layers, the MAE reduced to 33.73 meters. Comparing the baseline to the proposed FASTER R-CNN + LSTM method displays a substantial improvement, reducing the MAE by 64.7%.

Lastly, Figure 7 presents a visualization comparing the performances of FASTER R-CNN + LSTM and FASTER R-CNN + FFNN against the ground truth. Notably, both methods exhibit increased error in the areas with higher obstruction, indicated in red, due to the tree coverage that inevitably blocks the camera from viewing more landmarks. In contrast, regions with lower obstruction benefit from higher visibility, resulting in a 37.7% improvement in accuracy.

Future improvements could include the following:

- Our dataset, used for developing and experimenting with the localization methods, was relatively small for deep learning applications. A larger dataset would offer a greater variety of image features and improved generalizability and accuracy.
- Our results indicate that using FASTER R-CNN as the landmark detector achieved the highest accuracy, but it may be too slow for resource-constrained computing platforms. Higher computing power could allow us to maximize this approach's potential or adopt even more complex object detection and localization models.
- We used existing FASTER R-CNN and YOLO models for landmark detection. Custom architectures tailored to our specific use case could potentially enhance object detection accuracy and, in turn, localization results.

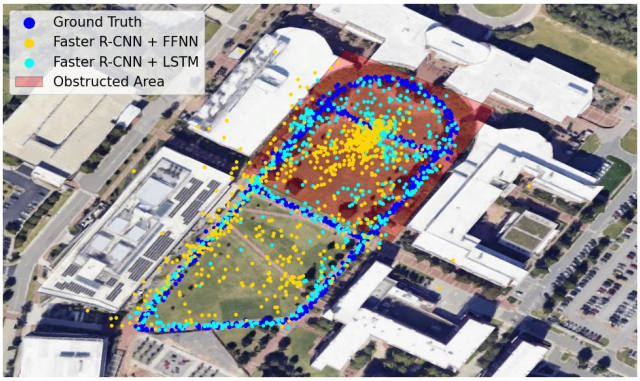

Fig. 7. Locations predicted by Faster R-CNN with LSTM vs. FFNN.

VI. CONCLUSION

In this paper, we presented a vision-based localization technique that utilizes deep learning models for mobile autonomous robot applications. Our proposed framework would eliminate the need for expensive, specialized localization sensors and leverage the inexpensive camera already present in autonomous ground vehicles. Landmarks of a target environment are detected from images captured and used to predict the vehicle's location. We implemented and evaluated combinations of object detection and localization models to assess the trade-offs between accuracy and execution efficiency. The results show that using FASTER R-CNN for landmark detection and LSTM for localization performed the best among the tested approaches, while YOLO with LSTM could also be more suitable for situations requiring higher update rates and/or lower computing power. Our prototype implementation on a custom-built mobile vehicle demonstrated superior performance compared to existing methods, validating the feasibility of the proposed approach.

REFERENCES

[1] "Meet roxo, the fedex sameday bot," https://www.fedex.com/en-ws/about/sustainability/our-approach/roxo-delivery-robot.html.
[2] "Serve robotics rolls out robotic delivery for uber eats," https://www.serverobotics.com/uber-eats.
[3] "The k5 asr," https://knightscope.com/products/k5.
[4] "Skydio x10," https://www.skydio.com/x10.
[5] "Gps accuracy," https://www.gps.gov/systems/gps/performance/accuracy/.
[6] K. Berntorp, "Joint wheel-slip and vehicle-motion estimation based on inertial, gps, and wheel-speed sensors," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 3, pp. 1020–1027, 2016.
[7] X. Kang, J. Li, X. Fan, and W. Wan, "Real-time rgb-d simultaneous localization and mapping guided by terrestrial lidar point cloud for indoor 3-d reconstruction and camera pose estimation," *Applied Sciences*, 2019.
[8] G.-S. Cai, H.-Y. Lin, and S.-F. Kao, "Mobile robot localization using gps, imu and visual odometry," in *2019 International Automatic Control Conference (CACS)*, 2019.

[9] "Emlid reach rs2+," https://e38surveysolutions.com/products/emlid-reach-rs2.

[10] H. Wang, C. Wang, C.-L. Chen, and L. Xie, "F-loam : Fast lidar odometry and mapping," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[11] "Helios series - robosense," https://store.robosense.ai/products/helios-series?variant=44365447692341.

[12] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *IEEE transactions on robotics and automation*, vol. 13, no. 2, pp. 251–263, 1997.

[13] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 736–749, 2015.

[14] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016. [Online]. Available: https://arxiv.org/abs/1506.01497

[15] S. Nilwong, D. Hossain, S.-i. Kaneko, and G. Capi, "Deep learning-based landmark detection for mobile robot outdoor localization," *Machines*, vol. 7, no. 2, 2019.

[16] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint detection and description of local features," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[18] M. Tyszkiewicz, P. Fua, and E. Trulls, "Disk: Learning local features with policy gradient," *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[19] L. Svarm, O. Enqvist, F. Kahl, and M. Oskarsson, "City-scale localization for cameras with known vertical direction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[20] B. Zeisl, T. Sattler, and M. Pollefeys, "Camera pose voting for large-scale image-based localization," *International Conference on Computer Vision (ICCV)*, 2015.

[21] F. Camposeco, A. Cohen, M. Pollefeys, and T. Sattler, "Hybrid scene compression for visual localization," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] S. Cao and N. Snavely, "Minimal scene descriptions from structure from motion models," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[23] P.-E. Sarlin, D. DeTone, T.-Y. Yang, A. Avetisyan, J. Straub, T. Malisiewicz, S. R. Bulo, R. Newcombe, P. Kontschieder, and V. Balntas, "Orienternet: Visual localization in 2d public maps with neural matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[24] A. Valada, N. Radwan, and W. Burgard, "Deep auxiliary learning for visual localization and odometry," 2018. [Online]. Available: https://arxiv.org/abs/1803.03642

[25] E. Brachmann and C. Rother, "Learning less is more - 6d camera localization via 3d surface regression," 2018. [Online]. Available: https://arxiv.org/abs/1711.10228

[26] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, "Semantic visual localization," 2018. [Online]. Available: https://arxiv.org/abs/1712.05773

[27] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient effective prioritized matching for large-scale image-based localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1744–1756, 2017.

[28] F. Cao, S. Wang, X. Chen, T. Wang, and L. Liu, "Bev-lslam: A novel and compact bev lidar slam for outdoor environment," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2462–2469, 2025.

[29] S. Wen, X. Li, X. Liu, J. Li, S. Tao, Y. Long, and T. Qiu, "Dynamic slam: A visual slam in outdoor dynamic scenes," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–11, 2023.

[30] F. Liu, Y. Ma, J. Wang, S. Wu, Y. Zhao, and Z. Wang, "Research on multi-feature constrained adaptive visual slam localization method based on decision tree," *IEEE Access*, pp. 1–1, 2025.

[31] M. D'arienzo, "An experimental comparison of basic device localization systems in wireless sensor networks," *Network 2025*.

[32] M. Kabiri, C. Cimarelli, H. Bavle, J. Luis Sanchez-Lopez, and H. Voos, "A review of radio frequency based localisation for aerial and ground robots with 5g future perspectives," 2024.

[33] G. Robles, J. Manuel Fresno, and J. Manuel Martínez-Tarifa, "Radio-frequency localization of multiple partial discharges sources with two receivers," 2018.

[34] S. J. Lee, D. Kim, S. S. Hwang, and D. Lee, "Local to global: Efficient visual localization for a monocular camera," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 2231–2240.

[35] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *International Conference on Learning Representations (ICLR)*, 2020.

[36] X. Dai, Y. Chen, J. Yang, P. Zhang, L. Yuan, and L. Zhang, "Dynamic detr: End-to-end object detection with dynamic attention," *International Conference on Computer Vision (ICCV)*, 2021.

[37] A. Shustanov and P. Yakimov, "Cnn design for real-time traffic sign recognition," *Procedia Engineering*, 2017.

[38] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight cnn," *IEEE International Conference on Edge Computing (EDGE)*, 2018.

[39] C. Liu, Y. Tao, J. Liang, K. Li, and Y. Chen, "Object detection based on yolo network," in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 2018.

[40] PyTorch. Faster rcnn resnet50 fpn0: Torchvision documentation.

[41] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[42] M. Yue, L. Zhang, J. Huang, and H. Zhang, "Lightweight and efficient tiny-object detection based on improved yolov8n for uav aerial images," 2024.

[43] Q. Liu, W. Huang, X. Duan, J. Wei, T. Hu, J. Yu, and J. Huang, "Dsw-yolov8n: A new underwater target detection algorithm based on improved yolov8n," 2023.

[44] N. Ma, Y. Wu, Y. Bo, and H. Yan, "Chili pepper object detection method based on improved yolov8n," 2024.

[45] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "Yolov6: A single-stage object detection framework for industrial applications," 2022.

[46] A. Sak Hasim, Senior and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *INTERSPEECH*, 2014.

[47] W. Lu, J. Zhang, X. Zhao, J. Wang, and J. Dang, "Multimodal sensory fusion for soccer robot self-localization based on long short-term memory recurrent neural network," *Journal of Ambient Intelligence and Humanized Computing*, 2017.

[48] A. Poulose and D. S. Han, "Uwb indoor localization using deep learning lstm networks," *Applied Sciences*, vol. 10, no. 18, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/18/6290

[49] D. Pang and X. Le, "Indoor localization using bidirectional lstm networks," in *2021 13th International Conference on Advanced Computational Intelligence (ICACI)*, 2021, pp. 154–159.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Microsoft Research*, 2014.

[51] X. Qi, Y. Wei, X. Mei, R. Chellali, and S. Yang, "Comparative analysis of the linear regions in relu and leakyrelu networks," *Communications in Computer and Information Science*, 2023.

[52] L. S. Karumbunathan. Nvidia jetson agx orin series: A giant leap forward for robotics and edge ai applications. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf

[53] "Zed 2i stereo camera," https://www.stereolabs.com/products/zed-2i.

[54] "Roboflow," https://roboflow.com.

[55] "Open street map," https://www.openstreetmap.org/.

[56] "How much distance does a degree, minute, and second cover on your maps?" https://www.usgs.gov/faqs/how-much-distance-does-a-degree-minute-and-second-cover-your-maps.

[57] "Nvidia geforce rtx 4090," https://www.techpowerup.com/gpu-specs/geforce-rtx-4090.c3889.

[58] "Gps basics," https://learn.sparkfun.com/tutorials/gps-basics.