Low-cost cloud-based HD-Map Updates for Infrastructure Management and Maintenance

Ahmed Mohamed[1*], Mohamed Elhabiby[2], Mohamed Moussa[1,2,3], Naser El-Sheimy[1]

[1] Dept. of Geomatics Engineering, University of Calgary, 2500 University Dr NW, Calgary, Alberta T2N 1N4 Canada – (ahmed.mohamed2, mohamed.moussa1, elsheimy)@ucalgary.ca
[2] Public Works Department, Ain Shams University, Cairo, Egypt –(mmelhabiby, mohamed-osama)@eng.asu.edu.eg
[3] Micro Engineering Tech. Inc., Calgary, Alberta, Canada - m.moussa@microengineering.ca

KEY WORDS: HD map update, Object detection, Depth estimation, Crowdsourced data, Low-cost smartphone sensors, Objects matching.

ABSTRACT:

Recently, HD maps have various merits for achieving the highest level of self-localization accuracy, keeping track of the state of the road infrastructure and maintenance, and providing an indication if any repairs are required. Therefore, it is essential to keep the HD map up to date. However, the process of updating the HD map is exorbitant because the HD map is created using expensive sensor setups, and updating the map frequently via these setups will be costly. In this paper, a full pipeline is proposed for updating the HD map via a crowdsourced dataset that is collected with low-cost smartphone sensors. Furthermore, an Android application is developed and installed on a smartphone to collect the raw data. Once the dataset is collected from the area of interest, it will be uploaded automatically to the cloud server that is connected to the HD map database. Then, object detection, depth estimation, and matching algorithms are triggered on the cloud server to keep updating the HD map database. The positions of the detected objects from the crowdsourced dataset are estimated by using fused outputs of deep learning models and the Global navigation satellite system (GNSS) of a smartphone and then compared with the objects in the HD map through matching algorithms. The proposed model is considered the first comprehensive pipeline approach for updating HD maps with high a cost-effective and efficient solution.

1. INTRODUCTION

The advent of autonomous vehicles has sparked a paradigm shift in the transportation industry, offering immense opportunities to enhance safety, convenience, and efficiency. High-Definition (HD) maps have emerged as a key enabler in this regard, providing detailed and accurate information about road networks and their surroundings. It is worth noting that researchers in (R. Liu et al., 2020) build an accurate 3D map that contains multiple attributes, such as semantic information. For reaching the optimum accurate navigation, the HD map should be always updated to be matched with the changes in real world.

In (Pannen et al., 2019), they proposed an approach for HD Map change detection and update based on filters. Lane markings and road edge geometry were utilized as primary landmarks to estimate the vehicle's pose through a series of landmark detections and matching. Additionally, they utilized two particle filters for estimating a vehicle position and detecting the changes in the HD map. While, in (Park et al., 2022), they utilized an object detection model, YOLOv3 (Redmon & Farhadi, 2018), for updating a 3D HD map. When the YOLOv3 model outputs the bounding boxes, the canter point of each bounding box is used for comparing it with the center of a 3D object in the HD map for taking a decision of updating or unchanged the HD map objects.

Similarly, (Heo et al., 2020) utilized an instance segmentation deep learning network for matching between a camera frame and an HD map mask that is projected from the HD map by a given camera location of the captured frame. The output from the instance segmentation model is a similarity score mask that indicates if any object of the HD map should be added, changed, or removed. Although the models (Heo et al., 2020; Pannen et al., 2019; Park et al., 2022) achieved an acceptable accuracy in updating HD maps, they lack leveraging of crowdsourced data from different contributions because the crowdsourced dataset provides a scalable solution for updating HD maps. As the number of data increases, the confidence and accuracy of updating HD maps with low-cost sensors increases. Additionally, crowded data provides an extensive dataset that is sufficient for a better representation of the real world.

The researchers in (Zhang et al., 2021) exploit the benefit of a crowdsourced dataset that is collected from a camera, Controller Area Network (CAN), and a high-end localization sensor, called SPAN-IGM-A1 that provides accurate positioning via integrating the reading from GNSS and Inertial Measurement Unit (IMU). They estimated the segmentation of static objects by utilizing the BiSeNet network (Yu et al., 2018) and then it is integrated with Simultaneous Localization and Mapping (SLAM) reading and the positioning sensors for predicting the point cloud of static objects. After that, the predicted point cloud data are denoised, clustered, and vectorized for easing matching it with HD map objects via Intersection Over Union (IOU) method. Even though they reached good accuracy in matching crowdsourced data with the HD map, their model is not affordable because they depend on expensive high-end sensors. In a corresponding manner, (Jo et al., 2018) employs a SLAM algorithm and particle filter to update an HD map using data from the onboard sensors of an autonomous vehicle, including the light detection and ranging (LiDAR) sensor. It should be noted that LiDAR sensors are generally regarded as expensive for implementation.

In (Cho et al., 2022), they tried to tackle the problem of utilizing expensive sensors for updating HD maps by using low-cost onboard sensors instead, which can be installed on taxis or buses for collecting crowdsourced datasets. The researchers in (Cho et al., 2022) focused mainly on updating road lanes alongside traffic signs and lights in Seoul, South Korea. They compared the crowdsourced dataset with HD map objects and clustered unassigned objects with the map to be utilized as new objects of the HD map. The main drawback of their model is focusing on one or two objects in updating maps without caring about other vital objects in roads like barriers or pedestrian bridges. The problem is overcome in (Kim et al., 2021) by proposing a general solution for updating multiple discrete landmarks and continuous landmarks, namely lanes. Their updating algorithm depends

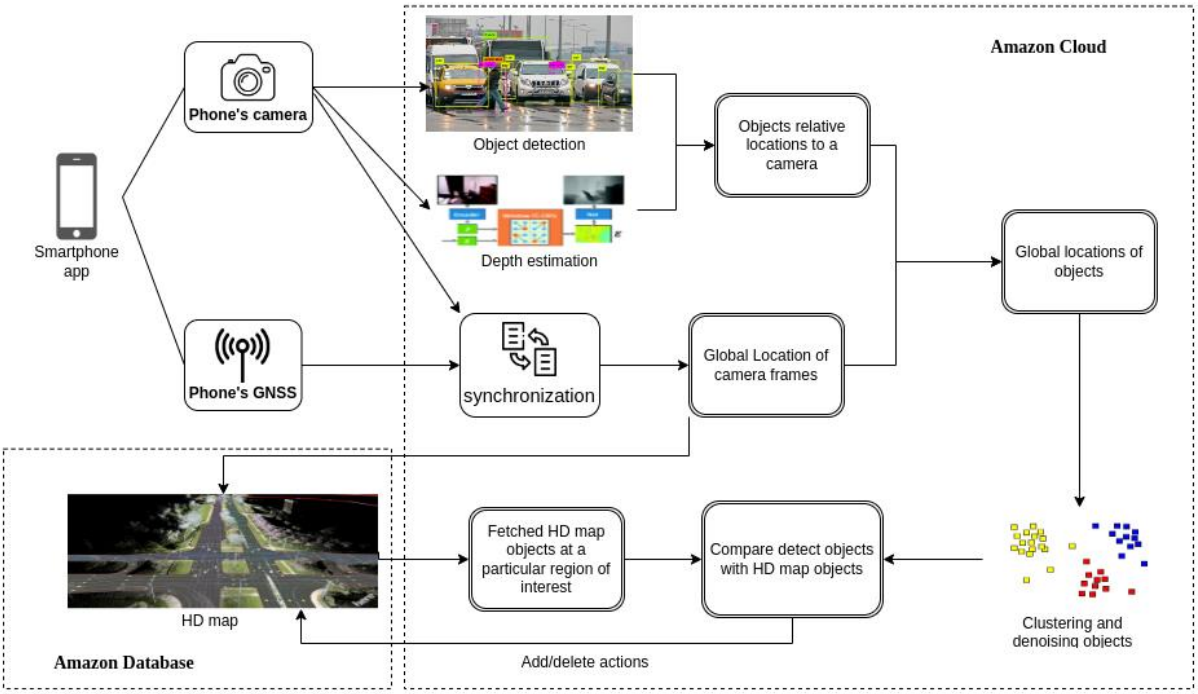

Figure 1. Full pipeline of HD map update with Low-cost smartphone sensors.

mainly on three criteria: the distance between a predicted object and an HD map object, the variance of the crowdsourced device's location, and the variance of detecting an object's location. Also, these criteria are utilized in clustering unassigned objects to the HD map to be classified as a new landmark to the map if the number of instances in a cluster exceeds a specific threshold.

One limitation observed in all models that utilize crowdsourced low-cost sensors for HD map updates is the absence of a comprehensive pipeline encompassing the entire update process. Each model tends to concentrate on a specific module of the update process, failing to provide a holistic view. As a result, the accuracy of HD map updates remains constrained. However, the proposition of a complete pipeline for map updates presents an opportunity for the research community and industries to achieve optimal accuracy by adjusting various parameters within the pipeline. By embracing and implementing such a comprehensive approach, significant advancements can be made in enhancing the accuracy of HD map updates.

In this research, the proposed comprehensive pipeline for updating HD maps is implemented by using crowdsourced low-cost smartphone sensors. Our research focused on updating various road classes in downtown Calgary, Canada, including barriers, traffic lights, pedestrian bridges, and various classes. Multiple navigation sensors are utilized as inputs for pipeline, such as Cameras, Lidar, GNSS, and IMU to create a highly accurate HD map covering the entirety of downtown Calgary. Then, an Android application is developed from scratch that connects low-cost smartphone sensors (phone's camera, GNSS, and IMU) to the cloud-based HD map database for perfect updates.

2. METHODOLOGY

In this section, we will provide a comprehensive overview of the complete process involved in updating HD maps using low-cost smartphone sensors through crowdsourcing. As depicted in Figure 1, the map update procedure initiates by streaming data from a smartphone's camera and GNSS sensors, which is collected by an Android app installed on a smartphone. Once the dataset is gathered from the app, it is uploaded to the Amazon cloud, where all the necessary deep learning models and algorithms for HD map updates, in addition to the HD map that is stored in the Amazon database.

Since the locations of the HD map objects are stored in a global format in the database, the predicted object locations from the smartphone also need to be global for effective comparison. To achieve this, the global locations of the detected objects are estimated through two simultaneous steps: predicting the objects' relative locations to a camera and calculating the global locations of the camera frames. Firstly, two deep learning models are applied to the camera frames of the phone: object detection and depth estimation. Notably, the You Only Look Once Version 5 (YoloV5) model (Jocher, 2020) and the Neural Window Fully-connected Conditional Random Fields (NeWCRFs) network (Yuan et al., 2022) are utilized for detecting landmarks of roads and predicting depth estimation from camera frames, respectively. By combining the outputs of these two models, the depth of the bounding boxes surrounding the landmarks is used to calculate the relative locations of the landmarks with respect to the camera. Concurrently, the global locations of camera frames are determined by synchronizing the timestamps of the camera frames and GNSS readings. This allows us to match the respective camera frames with the corresponding GNSS readings. Integrating these two steps provides the global locations of the landmarks. In order to decrease the computation time required for comparing HD map objects with detected objects, we employed clustering techniques to group objects with similar locations and subsequently eliminated the outlier object locations. Successively, the predicted clustered objects are compared with the HD map objects according to areas of interest. Based on this comparison, appropriate actions are taken on the

HD map objects, such as adding or removing objects from the map. In the following subsection, we will delve into a more detailed description of this pipeline.

2.1 Android Application

We created a brand-new Android app that gathers datasets from various sensors in a smartphone (camera, GNSS, IMU, and Magnetometer), as seen in Figure 2. It also retrieves data from a mobile network and calculates the smartphone's orientation angles relative to the North-East-Down (NED) coordinates. Furthermore, the app can connect to Onboard Diagnostics 2 (OBD2) through Bluetooth to obtain a vehicle's speed. All the collected data is saved simultaneously in easily readable Comma-Separated Values (CSV) files. Each file includes a timestamp for every frame of sensor readings to ensure synchronization. We collected camera data at 30 frames per second (FPS), GNSS readings at 20 FPS, and the phone's orientation angle whenever its position changed. Additionally, the app is linked to the Amazon cloud, automatically uploading the collected data to initiate HD map update algorithms. To accommodate various Android phones, our app is designed to work on multiple Android versions, starting from Android version 8.

Figure 2. The employed android application for data collection.

2.2 HD map

The Trimble MX9 device is responsible for generating the high-definition (HD) map. It comprises an MX9 laser scanner, a built-in Trimble GNSS-Inertial system, and four cameras, including a spherical camera with a field of view that covers 90% of the entire sphere, as seen in Figure 3. The laser scanner has a 360-degree field of view and can accurately detect objects within a range of 1.2 meters to 420 meters, with a precision of 2-5 millimetres. In our project, we utilized this system to scan all the roads in downtown Calgary, Canada. The data collection took place under various weather conditions and at different times, including both day and night, in order to create a precise HD map encompassing both static landmarks (such as barrier bridges) and dynamic landmarks (such as cones). Once the map underwent cleaning and processing, it was saved as shape files. Subsequently, we extracted the data from these shape files and stored it in the Amazon database to facilitate easy access within the Amazon cloud environment.

2.3 Objects' relative locations to a camera

To determine the relative positions of specific objects (road landmarks) with respect to the camera's location, we introduce object detection and depth estimation models. The object detection model identifies and categorizes landmarks within a camera frame. The centre point of each bounding box enclosing a landmark is then calculated as a representative location point.

Figure 3. Trimble MX9 device.

Next, the depth estimation model comes into play, estimating the distance between the centre point of each object and the camera being used. By utilizing the camera's intrinsic parameters, the central points are computed in 3D space relative to the camera's location using the following formula:

$$Z = Depth(i,j);\quad X = \frac{(j-c_x)}{f_x}\cdot Z;\ Y = \frac{(i-c_y)}{f_y}\cdot Z, \qquad (1)$$

Where i, j = the location of the central point of an object on a camera frame
X, Y, Z = the location of the central point in 3d space relative to the camera
c_x, c_y = the optical center of the camera in x and y axis
f_x, f_y = the focal length in pixels of the camera in x and y axis

Because the intrinsic parameters of the camera that is represented in the optical center (c_x, c_y) and the focal length (f_x, f_y) in the x-axis and y-axis of the camera frame is crucial for estimating the object's location, our application was developed to collect all intrinsic parameters of the camera along with gathering the camera's frames. By utilizing this information, as well as the central location of the bounding box on a frame (i, j), we are able to calculate the objects' relative location to the camera (X, Y, Z), as indicated in equation (1).

For reaching optimum accuracy of estimating the relative location, we utilized the state-of-the-art object detection model, YoloV5 model (Jocher, 2020), and monocular depth estimation model, NeW CRFs model (Yuan et al., 2022). Yolov5, an advanced real-time object detection model, is a direct descendant of the Yolo series, which originated with YoloV1 (Redmon et al., 2016) and drew heavily from Yolov4 (Bochkovskiy et al., 2020). The Yolo framework revolves around the concept of dividing an image into numerous smaller grids, wherein each grid signifies a prediction for object localization and its associated class probabilities. YOLOv5 takes inspiration from YOLOv4 but introduces its own modifications and optimizations, and the model is designed to be focused on a balance between speed and accuracy. The researchers of YoloV5 introduce a versatile architecture for object detection, comprising three primary layers: Backbone, Neck, and Head, in which each layer can accommodate multiple object detection models. They meticulously select models that contribute to their objective of achieving highly accurate real-time object detection. To this end, they employ the CSPNet model (Wang et al., 2019) in the Backbone layer, incorporating the SPP model (He et al., 2014) and PAN model (S. Liu et al., 2018) in the Neck layer, and leveraging the YoloV3 model (Redmon & Farhadi, 2018) in the Head layer, see Figure 4. This thoughtful combination of models aids YoloV5 in achieving its desired outcomes.

On the other hand, the New CRFs model heavily relies on the utilization of the Fully Connected Conditional Random Fields

(FC-CRF) algorithm for monocular image depth prediction. The FC-CRF algorithm, which is a probabilistic graphical model, captures the spatial relationships between adjacent pixels or regions in an image. Each node in the FC-CRF graph represents a random variable, while the edges between nodes signify the dependencies or relationships between these variables. Given that depth estimation is inherently a task that involves spatial connections, where the depth of one pixel is influenced by its neighbouring pixels, the FC-CRF algorithm is employed to model the intricate interactions and dependencies across the entire image. Consequently, the NeW CRFs model leverages this approach by incorporating the FC-CRF module into each part of the model's decoder. This allows for depth estimation at different resolutions, and subsequently integrates the depth images into a single accurate depth representation. In each FC-CRF, the image is divided into multiple separable windows, each containing fully connected nodes that represent a small patch of the input image. It's important to note that the nodes within a window are fully connected, while nodes across different windows are not connected, as illustrated in Figure 5. The introduction of window-based division serves the purpose of reducing computational complexity while maintaining the accuracy of depth estimation.

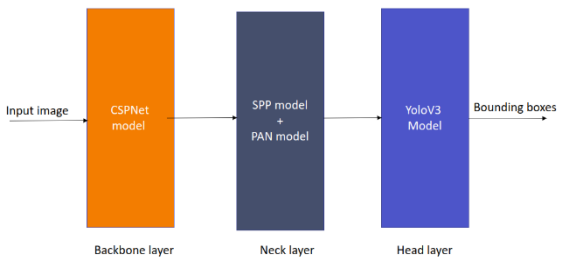

Figure 4. YoloV5 model architecture.

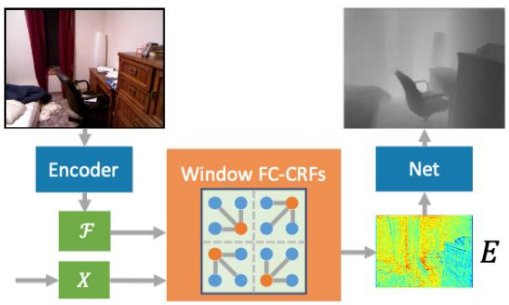

Figure 5. NeW CRFs model (Yuan et al., 2022) architecture.

2.4 Global Locations of objects

The estimation of real-world object locations involves combining two predictions: the global locations of camera frames and the relative locations of the objects with respect to the camera. The previous section provides a detailed description of the second prediction. For the global locations of camera frames, the synchronization of timestamps between a GNSS sensor and the camera is crucial for accurately estimating the position of each camera frame. Since timestamps are already recorded for each sensor's readings, as explained in section 2.1, predicting the locations of the camera frames becomes a straightforward task. It should be noted that the global location of a camera is calculated in Geodetic coordinates, namely latitude, longitude, and altitude. Therefore, the relative locations of the objects are mapped to the North-East-Down (NED) frame to seamlessly integrate them with the camera frames' locations in Geodetic coordinates. This mapping of relative locations from camera space to the NED frame involves two consecutive transformations: first, a transformation from camera space to the body frame, and then a transformation from the body frame to the NED frame. As depicted in Figure 6, the first transformation is relatively simple because the body frame closely resembles the camera space, except that the body frame's z-axis points outward while the camera's z-axis points inward and the x-axis direction of the body frame is the inverse of the x-axis direction of the camera space. The second transformation relies on the orientation angles of the phone with respect to the NED frame, which are already collected by the Android application as mentioned earlier. Finally, with knowledge of the frame location in Geodetic coordinates, the computation of the objects' locations in Geodetic coordinates will be performed by the method indicated in (Cai et al., 2011).

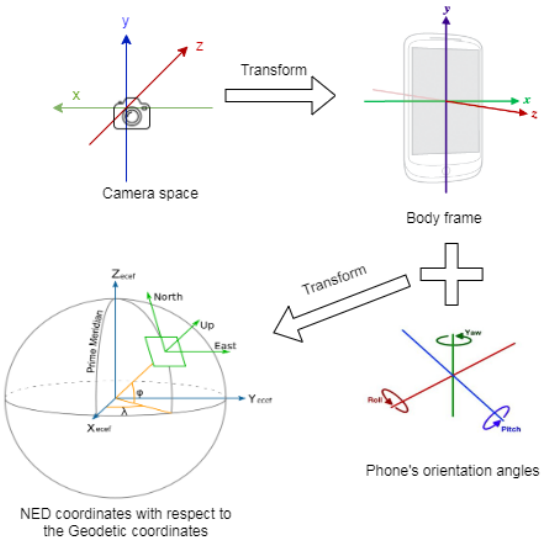

Figure 6. Transformation of an object's location from the camera space to the NED coordinates.

2.5 Compare HD map with crowdsourced objects

As we collected camera frames at a rate of 30 frames per second (FPS), there are often multiple objects with similar locations, as mentioned in section 2.1. To address this issue, we utilized clustering techniques to group the predicted objects and eliminate any outliers. Each cluster represents a single object, which is determined by calculating the average location of all objects within that cluster. Subsequently, we retrieved the HD map objects within the region corresponding to a group of camera frames` locations, in order to reduce computation time when comparing HD map objects with detected objects.

As illustrated in Algorithm 1, the comparison algorithm begins by extracting HD map objects from the chosen region, which are then added to the objects list if they belong to the same class as the detected object. Next, we select the HD map object that is closest to the detected object from the objects list. If the Euclidean distance between the selected HD map object and the detected object is below a distance threshold, the matched HD map object is retained. However, if there is no HD map object of the same class as the detected object or the distance between the selected HD map object and the detected object exceeds the distance threshold, the detected object is marked as unassigned. After that, all unassigned objects, gathered from different

crowdsourced devices, are clustered into multiple clusters, with each cluster represented by the average position of the objects within it.

We iterate over the clusters to find a cluster where the number of allocated objects surpasses another threshold known as the adding threshold. If a cluster meets this condition, the representative object of that cluster is added to the HD map, while disregarding clusters that do not meet the condition. Afterward, we calculate the number of non-matched occurrences for each HD map object that was fetched and added to the objects list within the same region where a detected object is in but did not match with any detected object. Finally, if the number of non-match occurrences for an HD object exceeds a third threshold referred to as the deleting threshold, the object will be removed from the HD map. It is worth noting that Algorithm 1 is implemented every time we have a group of camera frames, which is passed as input to the HD map for fetching HD map objects within the region of the frames' locations.

Algorithm 1: Comparing HD map objects with detected objects

Input: Detected objects for number of frames: *detObjs*,
HD map objects within a selected region: *hdObjs*
Output: New object candidates: *newObjs*,
Deleted object candidates: *delObjs*,
Normal object candidates: *normalObjs*

Initialization:
Set number of non-matched in each object in *hdObjs* to zero if the function is called for the first time.
Initialize *newObjs*, *delObjs*, *normalObjs* as an empty list
Define unassigned objects as an empty list: *unAssObjs*
Foreach *detObj* in *detObjs* **do**
    Search for the object classes in *hdObjs* that are the same as the class of *detObj* add the outputs to *selLs*
    Calculate the distance between *detObj* and every object in selLs
    Define the nearest object to *detObj as minSelObj*
    Define distance between *detObj* and *minSelObj* as *minDist*
    If *minDist < distance threshold* **Then**
        Add *minSelObj* to *normalObjs*
    Else
        Add *detObj* to *unAssObjs*
    End
End
Increment all non-matched objects in *hdObjs* by one
Cluster *unAssObjs* and add the outputs in *unAssObjsCls*
Foreach *cluster* in *unAssObjsCls* **do**
    Count number of objects in *cluster: cntCluster*
    Define the representative of *cluster* as: *repCluster*
    If *cntCluster > adding threshold* **Then**
        Add *repCluster* to *newObjs*
    End
End
Foreach *obj* in *hdObjs* **do**
    Define the number of non-matched for *obj* as: *nonCnt*
    If *nonCnt > deleting threshold* **Then**
        Add *obj* to *delObjs*
    End
End

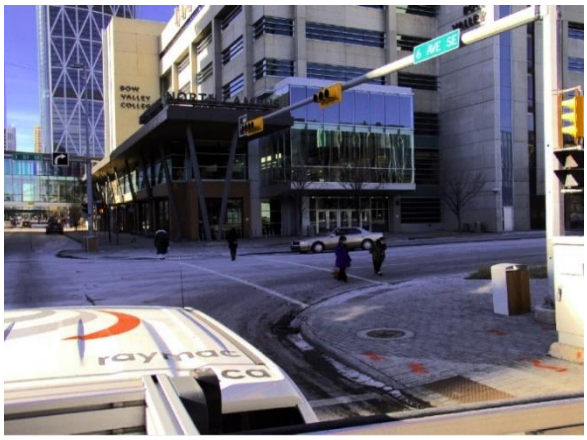

Figure 7. An image from our collected dataset by Trimble MX8.

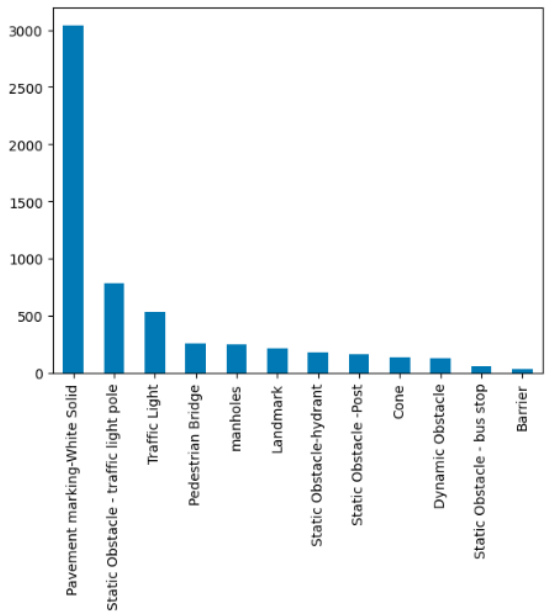

Figure 8. Bar plot of our collected dataset classes, where the horizontal axis represents class names and the vertical axis represents the number of instances in each class.

3. RESULTS

3.1 Training deep learning models

By utilizing our dataset collected in downtown Calgary using Trimble MX9, we performed training and testing on both YoloV5 and NeW CRFs models. Refer to Figure 7 for an image from our dataset. To train the object detection models, we labelled approximately 6,000 images from the collected dataset. This labelling process involved using an open-source labelling tool to annotate 12 classes of road landmarks, namely Pavement marking-white solid, Traffic light pole, Traffic light, Pedestrian bridge, Manholes, Dynamic obstacle, Hydrant, Cone, Post, Bus stop sign, Barrier, and Landmark (representing other road landmarks). Due to the imbalanced distribution of data classes, as shown in Figure 8, we employed data augmentation techniques and implemented a specialized data-splitting approach. In this approach, we ensured that the testing set had the same class distribution as the training set by selecting 30 consecutive frames from every successive 100 frames and incorporating them into the testing set. Additionally, we discarded bounding boxes with

a detection confidence probability below 0.3 to ensure the most accurate object detections possible.

For the purpose of training depth estimation, we utilized the MX9 laser scanner to calculate ground truth depth images, obtained by converting the estimated point cloud at each camera frame's pose. Because depth estimation plays a crucial role in the HD map updating algorithm, we specifically extracted reliable depth estimations within the range of 0.2 meters to 80 meters, disregarding values outside of this range.

3.2 Deep learning models assessment

The collected dataset was divided into a training set (70% of the total data) and a testing set (30%) to train both the YoloV5 and NeW CRFs models. To compare the inference times of the models, we conducted tests on an RTX 3070 GPU. As shown in Table 1, the YoloV5 model exhibited significantly lower inference times compared to the NeW CRFs model. Additionally, it was observed that the depth estimation process had a substantial impact on the overall time delay in updating HD maps. Consequently, the current depth estimation model is not suitable for real-time HD map updates and would require modifications to achieve real-time performance. However, this aspect is beyond the scope of this paper, as our current focus is on attaining accurate updates for HD maps.

Since the object detection model and the depth estimation model produce distinct outputs, we evaluated the accuracy of both models using different metrics. The YoloV5 model was assessed using the Average Precision (AP) metric, which approximates the area under the Precision-Recall (PR) curve. This was calculated by comparing the Intersection Over Union (IOU) of predicted bounding boxes with the ground truth bounding boxes, using varying IOU thresholds. To comprehensively evaluate the YoloV5 model, we employed a wide range of IOU thresholds, from 0.5 to 0.95, resulting in an AP of 0.166 within this range, as shown in Table 1.

On the other hand, the NeW CRFs model was evaluated using the Absolute Relative Error (ARE) metric, which quantifies the absolute difference between the true depth value and the estimated depth value, normalized by the true depth value. Our collected dataset yielded highly accurate depth estimation results for this model, with a precision of 0.052 in terms of ARE.

Model name	Inference time on RTX 3070	Utilized metric	Testing result
YoloV5	37ms	AP at IOUs from 0.5 to 0.95	0.166
NeW CRFs	199.59ms	ARE	0.052

Table 1. Comparison between YoloV5 and NeW CRFs model in the inference time and the metrics utilized for computing detection accuracy.

3.3 Evaluation of HD map updates

In order to assess the effectiveness of the proposed algorithm for updating HD maps, a dataset obtained through crowdsourcing was gathered along the fourth avenue of downtown Calgary. The selection of the downtown area for testing was based on the frequent alterations in landmarks and road infrastructure. Moreover, the accuracy of the phone's GNSS localization is hindered in this downtown location due to the urban canyon effect resulting from the presence of continuous buildings on either side of the roads. Consequently, we set out to prove accurate updates of our model even under these challenging conditions.

Our proposed algorithm proved the successful matching of detected objects by low-cost sensors with their corresponding landmarks in the high-definition (HD) map. As seen in Figure 9, one can see a minimal disparity between the detected landmarks and their corresponding counterparts in the HD map. Furthermore, our model effectively identifies new landmarks, such as cones, that are absent from the HD map at particular locations (refer to Figure 10). Additionally, the proposed model has the capability to estimate HD landmarks that are no longer present in the real world, as described in Algorithm 1, subsequently removing them from the map. Figure 11 demonstrates the accurate identification of a cone that is appropriately determined for removal from the HD map, as it remains undetected by any crowdsourced smartphones.

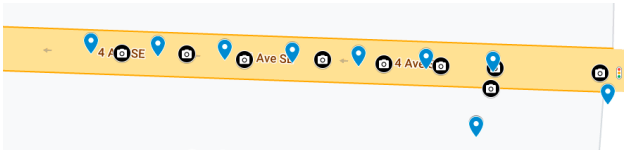

Figure 9. Visualization of matching detected landmarks with the HD map's landmarks, where camera markers and point markers represent the detected landmarks and the HD map's landmarks, respectively.

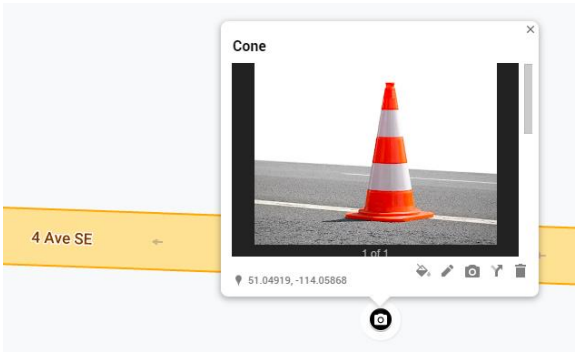

Figure 10. Location of the detected cone which will be newly added to the HD map.

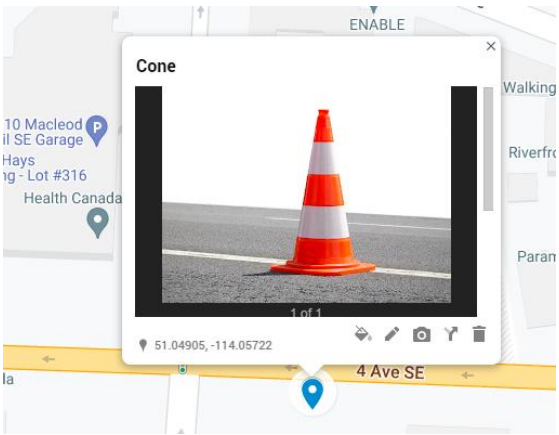

Figure 11. Location of the HD map cone that will be removed from the map due to not being detected by any crowdsourced sensors.

4. CONCLUSION

In this research paper, the proposed a comprehensive pipeline for is used to update HD maps by low-cost smartphone sensors. Our approach involved the development of an Android application to collect raw data, which was then automatically uploaded to a cloud server connected to the HD map database.

Once the data was uploaded, the object detection, depth estimation, and matching algorithms' outputs are fused in pipeline to update the HD map. By integrating deep learning models with the smartphone's Global Navigation Satellite System (GNSS) sensor for object position estimation, we were able to compare the detected objects from the crowdsourced dataset with the objects in the HD map. As a result, we could add new objects and remove existing ones to ensure the HD map accurately reflected real-life updates.

Our research presents a cost-effective and efficient solution for updating HD maps, leveraging the power of crowdsourced data and low-cost smartphone sensors. This approach allows for frequent updates to the HD map, enhancing its accuracy and maintaining its relevance to the evolving road conditions. Further improvements and optimizations can be explored to make the pipeline even more robust and capable of real-time updates, contributing to the advancement of HD map technologies and their applications in various domains.

ACKNOWLEDGMENTS

This research has been supported by funding of Prof. Naser El-Sheimy from NSERC CREATE and Canada Research Chairs programs.

REFERENCES

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (arXiv:2004.10934). arXiv. http://arxiv.org/abs/2004.10934

Cai, G., Chen, B. M., & Lee, T. H. (2011). *Unmanned Rotorcraft Systems*. Springer London. https://doi.org/10.1007/978-0-85729-635-1

Cho, M., Kim, K., Cho, S., Cho, S.-M., & Chung, W. (2022). Frequent and Automatic Update of Lane-Level HD Maps with a Large Amount of Crowdsourced Data Acquired from Buses and Taxis in Seoul. *Sensors*, *23*(1), 438. https://doi.org/10.3390/s23010438

He, K., Zhang, X., Ren, S., & Sun, J. (2014). *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition* (Vol. 8691, pp. 346–361). https://doi.org/10.1007/978-3-319-10578-9_23

Heo, M., Kim, J., & Kim, S. (2020). HD Map Change Detection with Cross-Domain Deep Metric Learning. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10218–10224. https://doi.org/10.1109/IROS45743.2020.9340757

Jo, K., Kim, C., & Sunwoo, M. (2018). Simultaneous Localization and Map Change Update for the High Definition Map-Based Autonomous Driving Car. *Sensors*, *18*(9), 3145. https://doi.org/10.3390/s18093145

Jocher, G. (2020). *YOLOv5 by Ultralytics* (7.0) [Python]. https://doi.org/10.5281/zenodo.3908559

Kim, K., Cho, S., & Chung, W. (2021). HD Map Update for Autonomous Driving With Crowdsourced Data. *IEEE Robotics and Automation Letters*, *6*(2), 1895–1901. https://doi.org/10.1109/LRA.2021.3060406

Liu, R., Wang, J., & Zhang, B. (2020). High Definition Map for Automated Driving: Overview and Analysis. *Journal of Navigation*, *73*(2), 324–341. https://doi.org/10.1017/S0373463319000638

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). *Path Aggregation Network for Instance Segmentation* (arXiv:1803.01534). arXiv. http://arxiv.org/abs/1803.01534

Pannen, D., Liebner, M., & Burgard, W. (2019). HD Map Change Detection with a Boosted Particle Filter. *2019 International Conference on Robotics and Automation (ICRA)*, 2561–2567. https://doi.org/10.1109/ICRA.2019.8794329

Park, Y.-K., Park, H., Woo, Y.-S., Choi, I.-G., & Han, S.-S. (2022). Traffic Landmark Matching Framework for HD-Map Update: Dataset Training Case Study. *Electronics*, *11*(6), 863. https://doi.org/10.3390/electronics11060863

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection* (arXiv:1506.02640). arXiv. http://arxiv.org/abs/1506.02640

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement* (arXiv:1804.02767). arXiv. http://arxiv.org/abs/1804.02767

Wang, C.-Y., Liao, H.-Y. M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019). *CSPNet: A New Backbone that can Enhance Learning Capability of CNN* (arXiv:1911.11929). arXiv. http://arxiv.org/abs/1911.11929

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). *BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation* (arXiv:1808.00897). arXiv. http://arxiv.org/abs/1808.00897

Yuan, W., Gu, X., Dai, Z., Zhu, S., & Tan, P. (2022). *NeW CRFs: Neural Window Fully-connected CRFs for Monocular Depth Estimation* (arXiv:2203.01502). arXiv. http://arxiv.org/abs/2203.01502

Zhang, P., Zhang, M., & Liu, J. (2021). Real-Time HD Map Change Detection for Crowdsourcing Update Based on Mid-to-High-End Sensors. *Sensors*, *21*(7), 2477. https://doi.org/10.3390/s21072477