Hybrid Mobile Vision for Emerging Applications

Nan Wu
George Mason University
nwu5@gmu.edu

Felix Xiaozhu Lin
University of Virginia
felixlin@virginia.edu

Feng Qian
University of Minnesota
fengqian@umn.edu

Bo Han
George Mason University
bohan@gmu.edu

ABSTRACT

While mobile applications have greatly benefited from 2D computer vision algorithms such as object detection and classification, there is limited research on exploring 3D vision that is enabled by the increasing availability of depth cameras and LiDAR scanners on mobile devices. In this paper, we propose a hybrid mobile vision system that intelligently combines 2D and 3D vision for improving the performance of emerging applications such as augmented and mixed reality and volumetric content analytics. Our research is motivated by and explores the key observation of the crucial latency-accuracy tradeoff between 2D and 3D vision. We present a research agenda with two principles for enhancing mobile vision stack, complementing 3D vision with its 2D counterpart by leveraging their diverse resource/accuracy profiles and processing 3D data (*e.g.,* point clouds) with 2D vision cues for mitigating the high computation and storage costs.

ACM Reference Format:
Nan Wu, Felix Xiaozhu Lin, Feng Qian, and Bo Han. 2022. Hybrid Mobile Vision for Emerging Applications. In *The 23rd International Workshop on Mobile Computing Systems and Applications (HotMobile '22), March 9–10, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3508396.3512876

1 INTRODUCTION

Breakthroughs in computer vision (CV), particularly deep neural networks (DNN) that significantly boost the accuracy of tasks such as object detection & classification, scene recognition, and semantic segmentation [2, 16, 32], have enabled various emerging applications including augmented and mixed reality (AR/MR), video analytics, and autonomous driving [6, 20, 22]. As a pivotal building block of the aforementioned applications, the run-time performance of those CV algorithms and DNN models, in terms of accuracy and latency (*i.e.,* execution/inference time), is of the utmost importance. Due to the high computation complexity of CV algorithms/models, most existing work splits the pipeline of their applications between resource-constrained end devices (*e.g.,* smartphones, AR/MR headsets, and surveillance cameras) and powerful cloud/edge servers [20, 22, 49, 50].

Recent advances in 3D data capturing techniques (*e.g.,* commodity depth cameras and LiDAR scanners) make the creation of volumetric content that consists of 3D models of the real-world surroundings feasible on commercial off-the-shelf (COTS) devices such as smartphones and tablets. The CV community has also developed efficient deep learning models for 3D object detection & classification (*i.e.,* generating 3D bounding boxes for objects of interest) using point cloud data as the input [29, 34]. The depth information in volumetric data offers unique opportunities for emerging mobile applications [42]. For example, we can determine the spatial relationship between virtual and physical objects to correctly handle occlusions and create realistic AR/MR systems with more semantic meanings [7], especially when users freely move in 3D space and cause dynamic changes of the occlusion relationship. Intelligent video analytics systems of volumetric content such as point clouds can leverage depth information for fall detection [39], the separation of occluded objects [48], accurate people counting [8], human behavior recognition [18], *etc.*

In this paper, we start with a pilot study that explores the strengths and weaknesses of 2D and 3D vision algorithms (§3). Our preliminary experimental results on a public dataset reveal a fundamental accuracy-latency tradeoff between 2D and 3D object detection & classification. While benefiting from the depth information in volumetric data, 3D vision offers higher accuracy than its 2D counterpart by detecting occluded and farther away objects in a scene, it consumes more computation resources and incurs a longer execution time (~10×). Moreover, volumetric data such as point clouds is large due to its 3D nature and its compression is compute-intensive owing to its sparsity and irregularity [12].

Motivated by the above observations, we propose a hybrid mobile vision system that strategically combines the strengths of 2D and 3D vision with two key principles. First, by leveraging the latency-accuracy tradeoff, we complement 3D vision with 2D visual cues for enhancing the efficiency of object detection & classification (§4.1). More specifically, we propose hybrid input data that explores diverse factors (*e.g.,* the distance of objects and potential occlusion of objects), hybrid vision tasks that fuse 2D objection detection and 3D object classification and reduce 3D vision workload with preprocessing using 2D vision, and hybrid computation locations that dynamically split the execution of 2D and 3D vision models between the client and the server. Second, we facilitate 3D volumetric data processing with 2D vision features to optimize the computation and storage overhead for volumetric content analytics, retrieval, and compression (§4.2). For example, we index volumetric content based on detected 2D bounding boxes to speed up its analytics tasks, accelerate on-device point cloud retrieval using 2D visual features, and improve the efficacy of volumetric content compression with inexpensive 2D vision primitives.

To the best of our knowledge, we are the first to propose a hybrid mobile vision system that effectively combines 2D and 3D vision to neutralize their weaknesses. Thus, our primary contribution is to bring to the attention of mobile application developers our key insights on the efficiency of dealing with different input data formats for various mobile vision tasks, especially object detection & classification and volumetric content analytics, retrieval, and

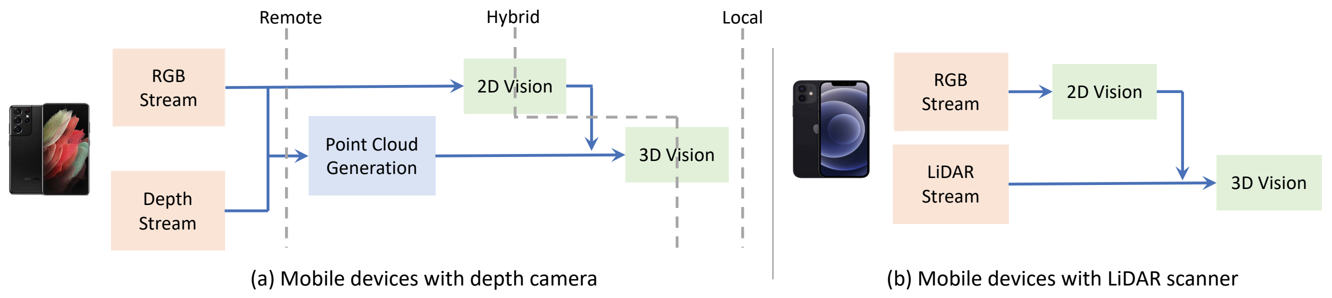

Figure 1: The system models for mobile devices equipped with a depth camera (a) or a LiDAR scanner (b). Point clouds will be generated directly from a LiDAR scanner or by combining RGB and depth streams. 2D/3D vision tasks and point cloud generation can be performed by either the mobile device (Local) or the server (Remote). The vision tasks can also be split between the device and the server (Hybrid).

compression. Our second contribution is to highlight the technical challenges of realizing such a hybrid mobile vision system.

2 SYSTEM MODEL

We show the abstracted system models of hybrid mobile vision in Figure 1 for devices with a depth camera or LiDAR scanner. For Android devices (*e.g.,* Samsung Galaxy S20 Ultra) that include a depth camera, we can generate point clouds as 3D models for surrounding scenes and objects from the RGB and depth streams. Given that this operation is usually compute-intensive, it is typically offloaded to a remote server (*e.g.,* at the network edge) for real-time mobile applications. Newly released iOS devices such as iPhone 12 Pro include a LiDAR 3D scanner that can directly create point clouds on-device, although the color information could not be captured by LiDAR. Thus, we still need to use the RGB stream to color the points when necessary.

As shown in Figure 1, to take advantage of the latency-accuracy tradeoff between 2D and 3D vision (§3), our proposed hybrid mobile vision utilizes the output of 2D vision to reduce the complexity of 3D vision for tasks such as object detection & classification and volumetric content analytics. 2D and 3D vision can be executed in different locations by considering different application scenarios and performance requirements, as explained in §4.1. We show three execution modes, remote, hybrid, and local, in Figure 1 (a), for mobile devices with a depth camera. The same partition methods apply to devices with a LiDAR scanner as well.

3 OBJECT DETECTION & CLASSIFICATION: 2D IMAGE VS. 3D POINT CLOUD

To motivate our proposed hybrid mobile vision, we compare the inference accuracy of the state-of-the-art deep learning models for 2D and 3D object detection & classification. We select the YOLOv4 [2] and PV-RCNN [34] models for 2D and 3D object detection and classification, respectively. We choose the widely-used KITTI Vision Benchmark Suite [9] that offers both 2D front-view camera images and volumetric LiDAR point clouds for the same scenes for an apple-to-apple comparison of the accuracy of those models. Using the KITTI dataset, we train the YOLOv4 model released by its authors and the PV-RCNN model released in the OpenPCDet [26] toolbox. OpenPCDet reports the accuracy of the 2D bounding boxes generated by projecting the corresponding 3D bounding boxes to the front-view images, which makes the comparison between YOLOv4 and PV-RCNN fair. Thus, we measure the detection accuracy of the projected 2D bounding boxes of PV-RCNN, instead of the original 3D boxes, and compare it with YOLOv4.

Table 1 reports the inference accuracy of YOLOv4 and PV-RCNN for three object classes, car, pedestrian, and cyclist. The KITTI dataset classifies objects using three difficulty levels, easy, moderate, and hard. The difficulty level is defined by considering the following metrics, minimum bounding box height in pixel (40, 25, and 25 pixels), maximum occlusion level (fully visible, partly occluded, and difficult to see), and maximum truncation ratio (15%, 30%, and 50%). The accuracy is calculated using the 40-point Interpolated Average Precision metric [37]. As we can see from this table, PV-RCNN outperforms YOLOv4, especially for the moderate and hard difficulty levels. The only exception is the easy difficulty level for pedestrians (71.35% vs. 74.53%). PV-RCNN performs better than YOLOv4 mainly due to its capability of detecting occluded and further away objects, thanks to the depth information provided in volumetric data. As shown in Figure 2, YOLOv4 fails to detect the car occluded by the cyclists and the cars that are further away in the scene. YOLOv4 cannot detect the jeep, probably due to the fact that there are only a few jeeps in the training set.

We next compare the computation resource utilization of 2D and 3D vision models. In the KITTI dataset, each point cloud contains ~130K points, on average. To make the comparison fair, we re-scale the 2D images to have a similar amount of pixels. Compared to YOLOv4, PV-RCNN consumes both more main memory (12.9% vs. 6.0%) and more GPU memory (2.1 vs. 1.6 GB). Moreover, the inference time of PV-RCNN is about 10× higher than that of YOLOv4 (525 vs. 44 ms for each inference).

Summary. The above preliminary results demonstrate that 3D vision on volumetric content offers better object detection and classification than 2D vision on images, at the cost of higher computation resource utilization. This observation motivates our proposed hybrid mobile vision. For example, we can first conduct 2D object detection on images to identify potential areas of interest, and then perform 3D object detection on points that belong to those areas to further improve the accuracy (*e.g.,* detecting occluded objects).

Model	Car			Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
YOLOv4 [2]	86.43	60.52	55.82	74.53	62.99	55.81	80.93	56.04	54.05
PV-RCNN [34]	97.78	93.38	91.60	71.35	64.63	61.07	93.59	77.12	74.14
Hybrid	97.91	93.40	91.63	69.59	62.77	58.02	93.08	76.98	72.84

Table 1: Accuracy of 2D and 3D object detection & classification models on the KITTI dataset [9]. The accuracy of PV-RCNN (3D model) is measured using the corresponding 2D bounding boxes projected on the same images for YOLOv4 (2D model). PV-RCNN performs much better than YOLOv4 for most categories. The last row shows the preliminary experimental results of our proposed hybrid mobile vision that leads to similar performance compared to PV-RCNN.

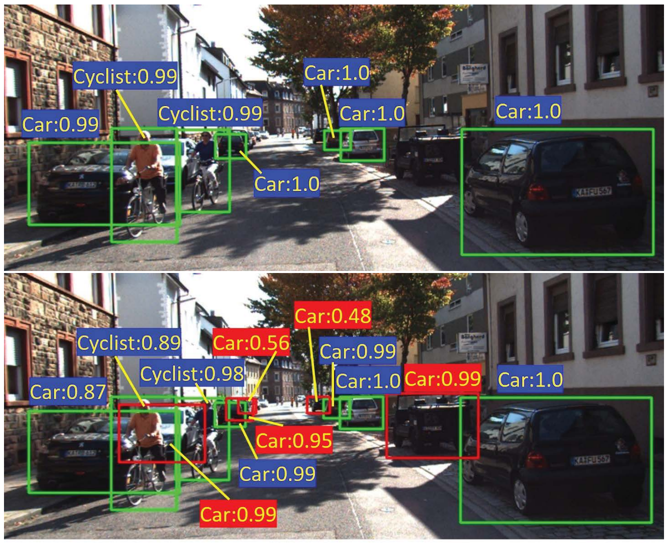

Figure 2: Comparison of object detection and classification, 2D (YOLOv4 on the top) vs. 3D (PV-RCNN at the bottom). PV-RCNN can detect more occluded and further away objects (with red labels) than YOLOv4.

4 RESEARCH AGENDA

While the 2D/3D vision synergy has been recently explored in robotics and autonomous driving [21], it still remains largely untapped for mobile applications. To enhance the mobile vision stack, we propose the following two principles. (1) For object detection & classification, we *complement* 3D with 2D vision by exploiting their diverse resource/accuracy profiles; (2) For volumetric content processing, we *enhance* 3D point clouds with 2D vision cues for mitigating the high computation and storage costs.

4.1 Enhancing 3D Mobile Vision with 2D for Object Detection & Classification

The measurement findings in §3 reveal the critical latency-accuracy tradeoff between 2D and 3D vision tasks such as object detection and classification. Balancing this tradeoff is particularly important on mobile devices that bear limited compute power and battery life. To realize this goal, we next describe strategies that combine 2D and 3D mobile vision in a principled manner.

Hybrid Input Data. This approach feeds separate input data to 2D vs. 3D models. The underlying idea is to use 2D vision to handle "easy" content and use 3D vision to tackle more challenging content. Based on our preliminary study, we have identified several factors that can facilitate such hybrid input data for mobile vision. *(1) Distance*: we can apply 2D vision to nearby content and 3D vision to content with a further distance. Nearby objects have more details appearing in the camera view, making their detection easier; the missing details for far-away objects can be compensated by 3D vision that improves the detection accuracy. Note that the distance information is provided by depth data. The depth information can be overlayed onto 2D images with R, G, B channels to reveal the distance of each pixel. *(2) Occlusion*: we can apply 3D vision to possibly occluded objects to improve the system performance, as motivated by the observations in §3. Specifically, we can first run lightweight 2D object detection that produces a series of bounding boxes as exemplified in Figure 2. The detected 2D bounding boxes indicate potential occlusions. For example, in the top subfigure of Figure 2, there is an undetected car that is occluded by the two cyclists detected by YOLOv4. We can thus feed *only* 3D data belonging to the 2D bounding boxes to 3D object detection models. *(3) Confidence Score*: similar to the procedure above, we can first run 2D vision, and then apply 3D vision to detected bounding boxes with low confidence scores.

Our recent work DeepMix [10] takes a different approach to effectively combine 2D images and depth data for facilitating real-time 3D object detection on mobile headsets. After getting the depth information of objects identified by a 2D detection & classification model, instead of applying 3D object detection, we leverage lightweight measurement to estimate the 3D bounding boxes of objects of interest. By doing this, we can drastically reduce the on-device computation overhead and make DeepMix suitable for mobile headsets.

Hybrid Vision Tasks. Another dimension of fusing 2D and 3D vision is to apply different vision tasks, whose synergy is expected to achieve the desired balance between accuracy and computation latency. Specifically, there are two opportunities that we can exploit. *(1) Detection vs. Classification.* In applications that require fine-grained object detection & classification, we can use 2D content for detection (*e.g.,* identifying the existence of an object) and 3D data for classification (*e.g.,* inferring the detected object is a car). Compared to directly applying 3D vision for fine-grained object detection & classification, this approach substitutes a significant amount of 3D vision overhead with a lightweight 2D vision workload, with (hopefully) little degradation of the overall classification performance. *(2) Preprocessing vs. Detection/Classification.* Another opportunity is to apply preprocessing (*e.g.,* ground removal and preliminary semantic segmentation) on 2D content. The result will then be overlaid onto volumetric content to reduce the 3D vision workload. For example, for the scene in Figure 2, if the objects of interest consist of pedestrians, cyclists, and vehicles, we can first perform ground detection using 2D semantic segmentation [24, 25, 36] to identify the ground, and then identify a space (using the depth channel) with up to a certain height above the ground as the candidate region for the objects of interest. Only data belonging to this region will be processed by 3D vision.

Hybrid Computation Locations. 2D and 3D computer vision tasks can be executed in different locations to balance the key latency-accuracy tradeoff. We consider three options as illustrated by different partition lines in Figure 1(a). *(1) Local Execution* where both 2D and 3D models are executed locally on mobile devices. In this scenario, we can leverage heterogeneous local compute resources including CPU, GPU, DSP, and NPU that may exhibit diverse performance for different model structures [19, 45]. *(2) Hybrid Execution* where 2D and 3D inferences take place on either mobile devices or a nearby edge node respectively. While selective offloading [17, 43] is not a completely new idea, a challenge here is to strategically partition the workload at the appropriate granularity (*e.g.,* model-level, layer-level, or frame-level partition). Another unique challenge is to observe the dependency of 2D and 3D tasks as described above (*e.g.,* 3D input is pruned/filtered based on the 2D output). *(3) Remote Execution* where both 2D and 3D tasks are performed on an edge or a remote cloud for computationally weak devices. This scenario brings a new challenge of potentially excessive network bandwidth for uploading both 2D and 3D data. On most mobile devices, the 2D image stream with (R, G, B) channels and the depth stream are generated by separate cameras with different resolutions and frame rates (Figure 1(a)), and the former can usually produce (2D) images with higher quality and/or frame rate than the latter. Given the high correlation between the two streams, it is beneficial to judiciously merge the output streams of both 2D and 3D cameras by considering the cross-stream compression opportunities. On other devices equipped with a LiDAR scanner that produces point clouds (Figure 1(b)), merging the 3D point cloud and 2D image streams is more challenging and requires more research. An alternative approach is to upload 2D and 3D streams separately, and use the 2D stream as cues to facilitate 3D stream compression, as to be elaborated in §4.2.

Additional Challenges. It is worth mentioning that the above three dimensions (hybrid input data, vision tasks, and computation locations) can be jointly exploited. Given the large decision space and a wide spectrum of dynamics (content, network/compute resources, and vision models), it may be highly challenging to establish an analytical framework to decide which hybrid approach(es) and their configurations/parameters to use. Instead, a promising direction is a learning-based framework that makes informed decisions of when, what, and how to fuse 2D and 3D vision.

4.2 Optimizing Volumetric Content Processing with 2D Vision Features

The principle of hybrid mobile vision also applies to the processing of volumetric content such as point clouds, which has important use cases. (1) Volumetric content analytics. As described in §2, point clouds are derived from RGB and depth streams and can be consumed by training or fine-tuning of 3D vision models for analytics tasks. The training recurs from time to time in order to adjust to scene changes and data distribution shifts and incurs computation overhead. (2) On-device retrieval for localization. Once constructed, point clouds can be distributed to mobile devices and stored as 3D maps. At run time, the mobile device matches a point cloud of its surrounding environment with stored point clouds for precise 6DoF localization [30]. (3) Compression of volumetric content. In video streaming, volumetric content should be compressed before delivery [12]. However, 3D data is known to be difficult to compress due to its large size and irregularity [33].

The challenges are twofold: (1) Point clouds are large, *e.g.,* tens of MB per frame if uncompressed. This situation is exacerbated by some recent VR apps storing large 3D scenes on devices, *e.g.,* 20M points in 200MB compressed [35]. While point cloud compression is vital, it is computationally expensive. (2) Random access patterns. For instance, training often requires to access specific object classes in point clouds; retrieval needs to search in a large repository of point clouds.

Indexing Volumetric Content by Projected Frustums. To speed up content analytics, *indexing* has been common wisdom, where "indexes" refer to early results that are computed at low cost and stored alongside the full-fledged data.

The principle can be applied to volumetric content with 2D vision as a new twist. As the server-side point cloud store ingests RGB and depth data, it reconstructs unlabeled volumetric content while avoiding analyzing the volumetric content for efficiency. Instead, the system periodically samples RGB images and executes object detection on the sampled 2D images as indexes. The 2D images may come from the same source as the volumetric content (*e.g.,* one RGB-D camera) or a separate source (*e.g.,* one LIDAR and one RGB camera). The detected 2D objects serve as indexes, which help filter point clouds for human analysts to inspect and label. Specifically, from the detected 2D bounding boxes, the system extrudes *viewing frustums* in the 3D space [28]. Hence, for a point cloud, our system builds index $I = \{\langle f_1, c_1\rangle, ..., \langle f_n, c_n\rangle\}$, where $f \subset R^3$ is a 3D viewing frustum and c is an object class. We can store 3D points keyed by viewing frustums and the remaining points as the scene background. Given an object class (*e.g.,* cyclists) requested in model training, our system can retrieve the point clouds in viewing frustums without loading unneeded points.

On-device Point Cloud Retrieval. A mobile device can localize itself by matching the point cloud of its surrounding environment with on-device 3D maps. The initial match is often the most expensive stage, where the device needs to quickly narrow down to a small set of candidate point clouds. Exhaustive search is slow, as it would decompress a large number of point cloud frames and run point cloud registration algorithms such as iterative closest point (ICP) for a pairwise match. The system may use GPS signal as geofence, which, however, is not always available, for example, in urban canyons where GPS signal is weak.

To speed up the initial match, 2D images offer rich visual cues, which can be extracted with inexpensive vision primitives. The resultant 2D features can be used for rough yet valuable estimation. Localization with 2D features (*e.g.,* SIFT) is well understood. When our system constructs point clouds for 3D maps, it also samples 2D images with their SIFT features. By aggregating the 2D features (*e.g.,* using bag-of-words), the system constructs global descriptors for individual point clouds. To localize, it collects both images and point clouds and uses the 2D descriptors to filter candidate point clouds for the match, therefore reducing the overhead.

Volumetric Content Compression with 2D Cues. Since volumetric content usually exhibits high data redundancy, it shall be compressed where it was captured or constructed. The efficacy of compression schemes, notably the compression ratio and speed, hinges on a good estimation of point cloud characteristics, such as camera motion and scene planes. Often, such information is extracted from the raw point clouds. To ensure information accuracy, the compressor periodically samples a point cloud stream as it is being captured. Doing so incurs substantial overhead because of the large data volume and irregularity. To this end, we can estimate compression parameters by processing 2D images captured near the same time, forgoing deep processing of point clouds for estimation. Fundamentally, 2D images are projections of point cloud to a lower-dimensional space; many point cloud properties needed for compression can be estimated by well-established 2D vision primitives.

The 2D cues can assist multiple compression schemes:

- *Cheap Plane Detection.* It is common for point cloud compression schemes such as G-PCC [33] in MPEG to store points as space partitioning trees such as Octree where leaf nodes represent points that fit in a 2D plane. The compressor can estimate planes from 2D images with low-cost segmentation operators and project planes to the 3D space.
- *Adapting Voxel Granularities.* Some compression schemes store a point cloud frame as spatial cells ("voxels") that are compressed individually [33]. The granularities can be heterogeneous across a frame, where "hotspots" regions such as foreground objects are covered with smaller voxels for higher spatial resolution, catering to deeper analytics. To this end, 2D vision can classify the scene and detect objects for hotspot regions.
- *Key Frame Selection.* A compressed point cloud stream consists of key frames and inter-coded frames in between; the inter-coded frames only store the difference between key frames. The intervals of key frames depend on the estimation of camera motions and scene changes. Without analyzing the point cloud, such estimations can be done by low-cost processing of 2D images such as optical flow, which guides the key frame selection for point cloud streams.

It is worth noting that compressing volumetric content with 2D cues differs from prior work of projecting volumetric content to 2D for compression. Such techniques project point clouds to 2D space using spherical projection [41] or orthogonal projection (V-PCC [33] in MPEG) and compress the 2D images. The information loss is significant, since they only store the resultant 2D frames in lieu of 3D point clouds.

5 PRELIMINARY EXPERIMENTAL RESULTS

To validate the effectiveness of our proposed hybrid mobile vision, we implement part of the hybrid input data scheme in §4.1 and evaluate its performance.

Our implementation includes the following. Given an input point cloud, we first use a pre-trained YOLOv4 2D object detection model to get the bounding boxes of detected objects on its corresponding 2D image. Since we have the camera parameters of the 2D image, we can project 3D points onto it and check whether a specific point falls in the detected 2D bounding boxes. By doing this, we can get not only points that map to detected objects in these 2D bounding boxes, but also points that belong to objects occluded by these detected ones. Besides facilitating the detection of occluded objects, we identify points that are 20 meters away from the origin of the input point cloud, where the 2D image is taken. The goal is to detect further away objects that appear to be small in the 2D image. We then construct a new point cloud by combining the points that are mapped to the detected 2D bounding boxes and those of remote objects. Finally, We perform 3D object detection using a pre-trained PV-RCNN model on this newly constructed point cloud, instead of the original one, which could potentially decrease the inference time due to the reduced number of points.

We report the preliminary experimental result of inference accuracy in the last row of Table 1, using the same KITTI dataset. Overall, the accuracy of our proposed hybrid input data scheme is similar or slightly worse than that of PV-RCNN, but outperforms YOLOv4 for most categories. More specifically, the inference accuracy of the hybrid scheme for cars is almost the same as PV-RCNN, and 11.48%, 32.88%, and 35.81% higher than that of YOLOv4. The hybrid inference accuracy for pedestrians is worse than YOLOv4 for the easy difficulty level (69.59% vs. 74.53%), but better than YOLOv4 for the hard difficulty level (58.02% vs. 55.81%). Finally, for the cyclist category, the inference accuracy of our hybrid input data schemes outperforms YOLOv4 for all three difficulty levels.

While the above implementation achieves comparable performance with PV-RCNN in terms of inference accuracy and performs better than YOLOv4 in general, its inference time is significantly reduced. As mentioned in §3, PV-RCNN takes about 525ms for each inference. It takes only around 419ms for the hybrid input data scheme, reducing the inference time by about 20%. The reason is that PV-RCNN's inference time can be drastically reduced when there are fewer points to process.

6 RELATED WORK

RGB-D based 3D Object Detection. While most existing schemes of 3D object detection leverage either point clouds [29, 34] or 2D images [4, 5] as input, there do exist schemes that combine RGB images and depth data for 3D object detection [28, 38, 40]. For example, DSS [38] proposes the first 3D region proposal network that takes a 3D scene from RGB-D data to generate a 3D object proposal and the first 2D+3D object recognition network that predicts the object label and regresses the 3D bounding box. F-PointNet [28] leverages 2D object detection to narrow down the 3D space and utilizes PointNet [29] to conduct segmentation on selected 3D frustums for estimating the 3D bounding box. To improve F-PointNet, Trans3D [40] proposes a 3D object detection network to help label 3D bounding boxes with 2D box labels, by transferring the information learned from classes that have both 2D and 3D bounding boxes. Although these approaches can reduce the amount of to-be-processed 3D data, their accuracy is usually not as good as point-cloud-based schemes.

Augmented and Mixed Reality. Mobile AR/MR depends on CV algorithms such as object detection, classification, recognition, and tracking [22, 50, 52]. For example, Jaguar [50] utilized GPU-accelerated edge to achieve accurate, low-latency object recognition and robust, context-aware object tracking. Liu *et al.* [22] improved the accuracy and latency of object detection for mobile AR by decoupling the rendering pipeline from edge offloading. Existing work also leveraged collaborations among multiple users for improving socialization [52], recognition accuracy [23], user experience [30], and scalability [51]. Existing work also leveraged collaborations among multiple users for improving socialization, recognition accuracy, and user experience [52]. More recent work on mobile AR/MR started to investigate the 3D effects of virtual content and the implications of 3D models on our daily life [30, 53]. For example, Xihe [53] is an edge-assisted framework that provided accurate omnidirectional lighting estimation using mobile 3D vision for AR applications. Our proposed hybrid mobile vision facilitates the understanding of 3D surroundings for mobile AR/MR applications by intelligently combining 2D and 3D object detection & classification.

Video Analytics. A typical video analytics pipeline [6, 20, 46, 49] consists of a data transmission scheduler and a vision processing engine. The scheduler determines what data will be sent to the server and how frequently to send these data. The decision is made by applying binary classification models, motion and object detection [49], or low-level vision features [20], all of which are typically lightweight. The back-end of those systems benefits from various deep-learning-based 2D object detection and classification models. For example, DSS [6] used Faster R-CNN [32], whereas AWStream [46] employed YOLOv2 [16]. Existing systems usually focus on one (or more) of the following optimization objectives, query latency (*e.g.,* Focus [13]), scalability (*e.g.,* Jain *et al.* [14]), server resource utilization (*e.g.,* VideoStorm [47] and Chameleon [15]), inference accuracy (*e.g.,* DDS [6]), and bandwidth consumption (*e.g.,* AWStream [46]). To the best of our knowledge, there is no existing work on video analytics of volumetric content. In this paper, we propose hybrid mobile vision that explores the accuracy-latency tradeoff between 2D and 3D vision to facilitate video analytics of volumetric content.

Video Storage. Existing systems focus on 2D visual data, such as Facebook's Haystack [1] and Intel's VDMS [11, 31] that store images rather than videos. NVIDIA's Video Loader [3] optimizes random loads of encoded video frames. Scanner [27] organizes video collections and raster data as tables and executes pixel-level computations in parallel. Our own work VStore [44] controls 2D video formats for queries. All these 2D video systems are inadequate, as they are oblivious to volumetric data structures and therefore incapable of coping with high data volume and computation cost. We are unaware of any prior analytics systems designed specifically for volumetric content.

7 CONCLUSION

In this paper, we proposed a research agenda for hybrid mobile vision that complements 3D vision tasks such as object detection and classification with its 2D counterpart and optimizes 3D data processing with 2D vision cues for reducing the computation and storage overhead. Our research is motivated by the observation that although 3D vision bears better accuracy than 2D vision, it leads to higher resource utilization and computation latency due to the 3D nature of volumetric data. Given the increasing popularity of depth cameras and LiDAR scanners on mobile devices, we hope our proposed research can inspire more work for advancing the mobile vision stack.

ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd Urs Hengartner for their insightful comments. The research of Bo Han and Nan Wu was funded in part by 4-VA, a collaborative partnership for advancing the Commonwealth of Virginia. Felix Xiaozhu Lin was supported in part by NSF awards #1846102, #1919197, #2106893, and the Virginia's Commonwealth Cyber Initiative.

REFERENCES

[1] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel. Finding a Needle in Haystack: Facebook's Photo Storage. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2010.
[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. https://arxiv.org/abs/2004.10934, 2020.
[3] J. Casper, J. Barker, and B. Catanzaro. NVVL: NVIDIA Video Loader. https://github.com/NVIDIA/nvvl, 2018.
[4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
[5] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo. Learning Depth-Guided Convolutions for Monocular 3D Object Detection. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[6] K. Du, A. Pervaiz, X. Yuan, A. Chowdhery, Q. Zhang, H. Hoffmann, and J. Jiang. Server-Driven Video Streaming for Deep Learning Inference. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, 2020.
[7] R. Du, E. Turner, M. Dzitsiuk, L. Prasso, I. Duarte, J. Dourgarian, J. Afonso, J. Pascoal, J. Gladstone, N. Cruces, S. Izadi, A. Kowdle, K. Tsotsos, and D. Kim. DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2020.
[8] C. Gao, J. Liu, Q. Feng, and J. Lv. People-flow Counting in Complex Environments by Combining Depth and Color Information. *Multimedia Tools and Applications*, 75:9315–9331, 2016.
[9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
[10] Y. Guan, X. Hou, N. Wu, B. Han, and T. Han. DeepMix: Mobility-aware, Lightweight, and Hybrid 3D Object Detection for Headsets. https://arxiv.org/abs/2201.08812, 2022.
[11] V. Gupta-Cledat, L. Remis, and C. R. Strong. Addressing the Dark Side of Vision Research: Storage. In *Proceedings of USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*, 2017.
[12] B. Han, Y. Liu, and F. Qian. ViVo: Visibility-Aware Mobile Volumetric Video Streaming. In *Proceedings of the Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2020.
[13] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying Large Video Datasets with Low Latency and Low Cost. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.
[14] S. Jain, G. Ananthanarayanan, J. Jiang, Y. Shu, and J. Gonzalez. Scaling Video Analytics Systems to Large Camera Deployments. In *Proceedings of International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2019.
[15] J. Jiang, G. Ananthanarayanan, P. Bodik, S. Sen, and I. Stoica. Chameleon: Scalable Adaptation of Video Analytics. In *Proceedings of Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.
[16] A. F. Joseph Redmon. YOLO9000: Better, Faster, Stronger. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[17] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. *ACM SIGARCH Computer Architecture News*, 45(1):615–629, 2017.
[18] H. Kim, S. Lee, Y. Kim, S. Lee, D. Lee, J. Ju, and H. Myung. Weighted Joint-Based Human Behavior Recognition Algorithm using Only Depth Information for Low-Cost Intelligent Video-Surveillance System. *Expert Systems With Applications*, 45:131—-141, 2016.
[19] R. Lee, S. I. Venieris, L. Dudziak, S. Bhattacharya, and N. D. Lane. MobiSR: Efficient On-Device Super-Resolution through Heterogeneous Mobile Processors. In *Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.
[20] Y. Li, A. Padmanabhan, P. Zhao, Y. Wang, G. H. Xu, and R. Netravali. Reducto: On-Camera Filtering for Resource-Efficient Real-Time Video Analytics. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for*

Computer Communication (SIGCOMM), 2020.

[21] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep Continuous Fusion for Multi-Sensor 3D Object Detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[22] L. Liu, H. Li, and M. Gruteser. Edge Assisted Real-time Object Detection for Mobile Augmented Reality. In *Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[23] Z. Liu, G. Lan, J. Stojkovic, Y. Zhang, C. Joe-Wong, and M. Gorlatova. CollabAR: Edge-assisted Collaborative Image Recognition for Mobile Augmented Reality. In *Proceedings of ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2020.

[24] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward Automatic Phenotyping of Developing Embryos from Videos. *IEEE Transactions on Image Processing*, 14(9):1360–1371, 2005.

[26] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[27] A. Poms, W. Crichton, P. Hanrahan, and K. Fatahalian. Scanner: Efficient Video Analysis at Scale. *ACM Trans. Graph.*, 37(4):138:1–138:13, July 2018.

[28] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[29] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] X. Ran, C. Slocum, Y.-Z. Tsai, K. Apicharttrisorn, M. Gorlatova, and J. Chen. Multi-User Augmented Reality with Communication Efficient and Spatially Consistent Virtual Objects. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2020.

[31] L. Remis, V. Gupta-Cledat, C. R. Strong, and M. IJzerman-Korevaar. VDMS: An Efficient Big-Visual-Data Access for Machine Learning Workloads. *CoRR*, abs/1810.11832, 2018.

[32] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

[33] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

[34] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[35] N.-J. Shih, P.-H. Diao, Y.-T. Qiu, and T.-Y. Chen. Situated AR Simulations of a Lantern Festival Using a Smartphone and LiDAR-Based 3D Models. *Applied Sciences*, 11(1), 2021.

[36] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[37] A. Simonelli, S. R. Bulò, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling Monocular 3D Object Detection. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[38] S. Song and J. Xiao. Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[39] E. E. Stone and M. Skubic. Fall Detection in Homes of Older Adults Using the Microsoft Kinect. *IEEE Journal of Biomedical and Health Informatics*, 19(1):290–301, 2015.

[40] Y. S. Tang and G. H. Lee. Transferable Semi-Supervised 3D Object Detection From RGB-D Data. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[41] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda. Compressing Continuous Point Cloud Data Using Image Compression Methods. In *Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[42] Z. Xie, X. Ouyang, X. Liu, and G. Xing. UltraDepth: Exposing High-resolution Texture from Depth Cameras. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2021.

[43] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu. DeepWear: Adaptive Local Offloading for On-Wearable Deep Learning. *IEEE Transactions on Mobile Computing*, 19(2):314–330, 2019.

[44] T. Xu, L. M. Botelho, and F. X. Lin. VStore: A Data Store for Analytics on Large Videos. In *Proceedings of EuroSys Conference*, 2019.

[45] J. Yi, S. Choi, and Y. Lee. EagleEye: Wearable Camera-based Person Identification in Crowded Urban Spaces. In *Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2020.

[46] B. Zhang, X. Jin, S. Ratnasamy, J. Wawrzynek, and E. A. Lee. AWStream: Adaptive Wide-Area Streaming Analytics. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, 2018.

[47] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live Video Analytics at Scale with Approximation and Delay-Tolerance. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[48] S. Zhang, C. Wang, S.-C. Chan, X. Wei, and C.-H. Ho. New Object Detection, Tracking, and Recognition Approaches for Video Surveillance Over Camera Network. *IEEE Sensors Journal*, 15(5):2679–2691, 2015.

[49] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee. The Design and Implementation of a Wireless Video Surveillance System. In *Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[50] W. Zhang, B. Han, and P. Hui. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proceedings of ACM Multimedia*, 2018.

[51] W. Zhang, B. Han, and P. Hui. SEAR: Scaling Experiences in Multi-user Augmented Reality. In *Proceedings of IEEE Conference on Virtual Reality and 3D User Interfaces (IEEE VR)*, 2022.

[52] W. Zhang, B. Han, P. Hui, V. Gopalakrishnan, E. Zavesky, and F. Qian. CARS: Collaborative Augmented Reality for Socialization. In *Proceedings of International Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2018.

[53] Y. Zhao and T. Guo. Xihe: a 3D vision-based lighting estimation framework for mobile augmented reality. In *Proceedings of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2021.