Multi Object Tracking and Panoptic Segmentation in Monocular Birds Eye View Images

Mircea Paul Muresan
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania Mircea.Muresan@cs.utcluj.ro

Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania Sergiu.Nedevschi@cs.utcluj.ro

***Abstract* — Bird's Eye View (BEV) maps have gained popularity in the autonomous driving field due to their ability to offer an information-rich and easily interpretable representation of the environment, playing a crucial role in various tasks such as motion planning, perception, and sensor fusion. In this paper, we propose a novel framework that integrates deep learning with grid filtering and feature engineering to achieve multi-object tracking and panoptic segmentation. The framework also validates object detections and removes spurious instances, resulting in a more robust environmental representation that includes tracked objects and semantic segmentation in both perspective and BEV domains. Furthermore, we introduce an original hybrid data association function that combines deep learning-based and handcrafted features to enhance object tracking accuracy. To support this approach, we present a dedicated dataset designed for training the deep learning model used in data association comprising four object categories: cars, pedestrians, riders, and animals. The proposed solution, implemented in C++, is integrated with contemporary deep learning frameworks and has been evaluated on international benchmarks such as Cityscapes and KITTI.**

Keywords—MOTS, BEV, tracking, panoptic segmentation, data association

I. INTRODUCTION

Autonomous systems such as cars, humanoids or other robotic platforms rely on a comprehensive understanding of the environment to perform essential tasks such as motion planning, obstacle avoidance, and sensor fusion [1]. A widely adopted approach for scene understanding is to represent the environment using Bird's Eye View (BEV) maps, which offer a compact and efficient way to encode the spatial layout of a scene [2], [3]. These maps are particularly advantageous due to their ability to provide a top-down depiction of the surroundings, that is easy to visualize, process, and interpret. The fact that in many scenarios, the critical information required for navigation is primarily concentrated on the ground plane makes BEV representations well-suited for downstream tasks such as trajectory estimation and behavior prediction [4]. Several state-of-the-art methods have demonstrated the effectiveness of deep learning in generating BEV representations from monocular or multi-view images, incorporating both semantic and instance-level segmentation to enhance scene understanding [5]. However, while panoptic segmentation in BEV has gained significant traction, current methods primarily focus on segmenting the scene [9], or detecting objects [6], few methods address panoptic segmentation, and object tracking is often overlooked, despite its importance for reliable perception in dynamic environments.

In this work we aim to overcome this limitation by proposing a framework that outputs the panoptic image containing tracked object instances. Existing methods use a variety of techniques to generate BEV maps from monocular images. Some approaches, such as IPM [7], use a homography to transform the perspective image to the bird's eye view, even though the flat earth assumption used by IPM-based methods affect their performance, in regions that lie above the ground plane. Other methods remove the ground plane using a deep learning approach and perform an IPM on the remaining objects, detecting them using a probabilistic occupancy grid and then tracking them using a particle filter [8], but not providing any multi class semantic segmentation information of the scene as output. Semantic BEV maps can also be generated using multiple sensors such as LiDARs and cameras, however such a pipeline requires information from multiple sensors and thus increases the cost of the whole system[9]. The multi object tracking (MOT) and segmentation tasks are intertwined in real world applications and by leveraging their synergies the task of decision-making in self-driving cars is improved. Panoptic segmentation can improve the overall object tracking performance, because it can offer more context information in the form of semantic data, that can be used in object association, and more precise information about object position, even in occluded scenarios, through the instance masks. Due to its real time performance, the tracking by detection paradigm has been employed in many state of the art MOT applications [10], [11].

When tracking multiple objects, state estimation becomes challenging due to the need to correctly associate each measurement with the corresponding tracked object, an issue commonly referred to as the data association problem. An incorrect assignment of a measurement can result in inaccurate parameter estimation or, in the worst case, cause the tracking system to misidentify targets. Data association performance can be affected by two main issues which are known in the literature [12] as the origin uncertainty and the motion uncertainty. The origin uncertainty refers to the fact that there is no prior knowledge of how current sensor readings relate to the previous ones, while the motion uncertainty refers to the fact that real world objects can have diverse moving patterns which should be considered when trying to associate the tracks and the detections. Effective tracking performance requires addressing both components of the data association problem. It is also important to note that after projecting objects in BEV space certain features that can normally aid data association, and are

visible from the perspective view, like the object appearance, are lost.

To address the issues mentioned above in this paper we propose the following contributions:

- We propose an original pipeline that combines panoptic segmentation, monocular depth estimation and object tracking to obtain a solution that tracks objects in birds eye view space. Moreover, the tracked instances—represented at both bounding box and mask level—can be accurately reprojected into the perspective view when required.
- We propose an original grid filtering approach that is applied on 3D points obtained from monocular depth perception points coming from object instances to clearly delimit objects in BEV space.
- We present a novel data association function that combines feature engineering with the output of Siamese networks for an improved data association.
- A dataset used for training the Siamese neural networks that contain 4 types of objects: cars, people, animals and riders is presented
- Finally, we present the implementation of a validation module, used for eliminating unwanted detections, and we also present the modified CNN architecture of one of the models used in our solution to enable export and execution in C++ using the TorchScript format.

The proposed solution has been evaluated on sequences from international benchmarks such as Cityscapes [33] and KITTI [34] [35]. The rest of the paper is organized as follows: in Section 2 a literature review of the state of the art in the field of bird's eye view detection and object tracking is presented. Section 3 describes each step of the proposed solution in detail. Section 4 presents the experimental results and finally, in Section 5 we conclude the paper.

II. Related Work

A. Detection and Segmentation in BEV

Bird's Eye View (BEV) perception has emerged as a fundamental technique in autonomous driving, enabling key tasks such as collision avoidance, path planning, and object tracking, by providing a unified and structured representation of the driving environment. Previous research has explored BEV perception from multiple perspectives, including camera-based methods, LiDAR-based approaches, and sensor fusion techniques.

Camera-based BEV (Bird's-Eye View) perception methods aim to transform perspective images into top-down representations. Some approaches use explicit geometric transformations based on depth estimation from monocular or stereo images, followed by reprojection into a unified 3D space [13]. LiDAR-based methods leverage 3D point clouds for accurate scene understanding, while voxel-based approaches divide the space into 3D grids and apply convolutions to extract spatial features [14]. To improve efficiency, pillar-based methods collapse the height dimension and apply 2D convolutions on BEV maps [15].

Multi-sensor fusion methods combine data from multiple (homogeneous or heterogeneous) sensors, extracting and aligning features into the BEV domain. MVX-Net [16] introduces PointFusion and VoxelFusion to enable early-stage modality interaction for 3D object detection. EPNet [17] improves fusion robustness using point-wise enhancement and a confidence-alignment loss. BEVFormer [18] employs a spatiotemporal transformer for integrating multi-view spatial features and past BEV representations, enhancing object detection and map segmentation.

VPN [19] addresses BEV semantic segmentation using frontal images and CARLA-simulated data but lacks effective use of geometric cues. In contrast, [9] proposes a two-stage approach combining monocular depth estimation and semantic segmentation to project a semantic point cloud into BEV, enabling better depth reasoning and adaptability across environments. Finally, [20] introduces the first BEV panoptic segmentation framework from frontal views, using dual transformers for vertical and planar mappings and a sensitivity-aware pixel weighting strategy.

B. Multi-Object Tracking and Segmentation

In tracking-by-detection frameworks, similarity between detections across frames is commonly assessed using object appearance and motion cues. The literature outlines three main strategies for data association: handcrafted feature engineering [21], data-driven methods [22], and hybrid approaches that combine both [26]. In [21], a handcrafted cost function is proposed, integrating features such as object size, color histograms, and motion via L2 distance, with optimal assignments solved using the Hungarian algorithm [24].

The authors from [22] extend Mask R-CNN [25] by adding an association head to their model, which learns feature embeddings for object identification. The approach is called TrackR-CNN and it works on color images integrating detection, tracking and segmentation in a single network. The authors in [26] combine feature engineered cost functions with a small CNN to create a data association function able to track objects in thermal images. The authors used features such as uniform binary patterns and created an original feature that combined the histogram of oriented gradients with local binary patterns to capture textures that can appear on objects from thermal images. The approach works in real time and can track objects at bounding box level. In [23] the authors use a feature engineered data association function and multiple motion models to track objects at bounding box and instance mask level. The authors in [27] propose single object BEV tracking solution that models target motion in BEV using convolutional layers and a simple regression head, followed by a global max-pooling and an MLP. The approach relies on motion modeling rather than explicit appearance matching. By leveraging voxel-based feature extraction and BEV spatial fusion, it efficiently tracks objects with diverse motion patterns at real-time speeds. In [8] the authors perform an object tracking in BEV using a particle filter approach. The objects are first segmented using an occupancy grid, and then the cells corresponding to each object are identified, clustered and a unique id is assigned to each cluster.

In our approach we build upon the state of the art by creating a novel approach that can track objects using a hybrid data association function and can perform a panoptic segmentation of the scene using a combination of deep learning techniques and grid filtering, where the tracked objects can then be displayed in BEV and perspective formats.

III. Proposed Solution

A. Processing pipeline overview and object validation

Section III describes the details of the implemented framework that performs panoptic segmentation and track objects at bounding box, instance and in the perspective and BEV spaces.

The framework processes monocular RGB images through multiple processing stages as illustrated in Figure 1. Since the proposed solution uses optical flow, which requires two frames to run, the main processing steps start after the second frame is received. Once the frame is received it is fed into a series of neural network models such as Efficient PS[36], that computes the panoptic segmentation, YoloV8x that detects instance masks and object bounding boxes, depth anything v2 [37] that estimates the relative depth to each pixel, which is then scaled to obtain the absolute distances and RAFT [38] dense optical flow. The monocular depth estimation module also receives the semantic segmentation map from the Panoptic Segmentation module and creates an enhanced point cloud that contains for each 3D point the semantic information as well as the RGB information from the original image. The instances detected by EfficientPS and Yolo are fed to a validation module, together with depth information and the tracked objects computed in the previous frame (if they are available). The validation module outputs a list of filtered objects that are tracked by the object tracker.

Using features extracted from the perspective image, objects are tracked and combined with the enhanced semantic point cloud. This combined data is then processed by a filtering and projection module, resulting in a BEV-space representation as the final output. The presented pipeline has been implemented fully in C++ using LibTorch and Onnx libraries in the case of depth anything v2. It is worth mentioning that the EfficientPS neural network model used for panoptic segmentation, as provided in the Git repository of the authors, cannot be exported in TorchScript format, compatible with C++. Modifications were necessary because layers such as In-Place Activated Batch Normalization cannot be traced in TorchScript, they were replaced with standard Batch Normalization layers followed by an ELU activation function to maintain result quality.

Another limitation of the TorchScript format is that it does not allow custom classes as input or output data for modules, permitting only tuples of tensors or, in specific cases, dictionaries. The instance head in Detectron2 uses objects of the class detectron2.structures.Instances for both, input during training and output during inference. To avoid export errors, the Detectron2 library provides the TracingAdapter class, which flattens custom classes by converting them into tuples of tensors. This library was used to successfully trace the model into TorchScript format. Moreover, the size of the intermediate channels was reduced using 1x1 convolutions and pre-processing and post-processing steps were also implemented and optimized in C++ to achieve the desired results. We empirically observed that the modifications introduced did not significantly affect the EfficientPS model accuracy, with performance variations under 1% on key metrics.

Panoptic and instance segmentation models are prone to errors such as missed detections, false positives, and misclassifications. Since the tracker focuses on traffic-related objects within 100 meters in depth and 30 meters laterally from the ego vehicle, a validation module is required to filter irrelevant or erroneous detections. Objects beyond 100 meters are excluded due to the increasing sparsity and unreliability of the point cloud. To ensure accurate environmental representation, the validation module cross-verifies detections using multiple redundant sources, especially in uncertain cases.

In our approach, we are using the information that comes from the two models (EfficientPS and Yolov8) and the third source represents the predictions (in the form of bounding boxes) coming from the tracked objects when available. In the case one object is detected by only one of the models, we verify whether the detection confidence and class probability exceed the 80% threshold, in which case we add the object to the validated object list. When an object is detected by both sources, several validation scenarios are considered and handled accordingly. In the scenario in which we do not have object predictions coming from a tracking module, we compute the intersection over union between the bounding boxes coming from the two models that analyze the scene. When the intersection over union has a good enough overlapping score, detected empirically as being over 70%, we check to see if the semantic classes are the same. If the semantic classes are the same the object is validated and placed in a validated objects list. Otherwise, we check to see which of the two detections has the higher-class probability score, and the detection having the higher score is kept provided the class probability is higher than 70%. In case the class probability is not higher than the previously mentioned threshold, the object is not validated. However, when the tracking predictions are available, in case

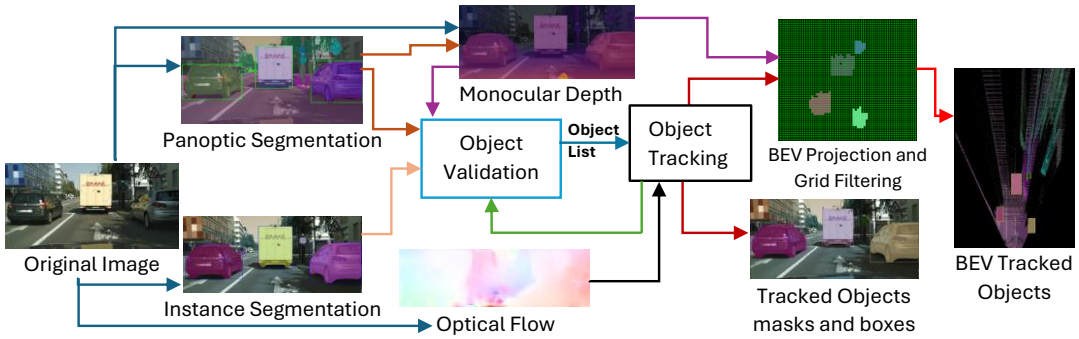

Figure 1. Pipeline overview. From an input RGB image (or pair), panoptic and instance segmentation, optical flow, and monocular depth are computed in parallel. Validated objects are obtained via an object validation module, then tracked and projected into a depth-based grid, resulting in a bird's-eye view representation.

there is an uncertainty regarding an object, we perform an additional check using the bounding boxes and the semantic class that is embedded in the predicted tracked object.

Each tracked object contains a histogram of semantic classes that is updated after each successful data association of a track and a detection, using the approach from section III B. The semantic class having the most votes from that histogram is considered the dominant semantic class of the track. We perform the intersection over union between the tracked object and detections as it was realised using the bounding boxes of two models. For the objects for which the intersection is above the previously specified threshold, we check the semantic class using the data from the tracked object and verify which class receives 2 of the 3 votes. The class having 2 of the three votes is considered the semantic class of the object, and the object is validated. Objects that do not satisfy the mentioned criteria are not included in the validated object list.

After the validation step, the validated object list is filtered and only objects that have the closest object point with respect to the ego vehicle at a distance up to 100 m in depth (Z axis) and 30m on the sides (X axis) are kept and the rest are removed. For each object instance in the validate object list, we obtain the closest point to the ego vehicle, traversing the object instance mask and computing for each point the Euclidean distance between the 3D point from the depth map (associated to that position) and the position of the ego vehicle. The depth information for each instance is obtained from depth anything v2 model, that is scaled to obtain the metric depth using the approach presented in [40]. After validation the object list is passed to the object tracker.

B. Data association and Tracking

The proposed tracking solution incorporates an original approach to computing the similarity cost, combining elements from three sources: information from dense optical flow, similarity from a Siamese network, and manually engineered features. Additionally, the approach includes a scoring system for both motion similarity between a tracked object and a detection, as well as an appearance score. The motion score consists of several components. First, the Euclidean distance between the tracked object and possible detections within the track's covariance ellipse is used.

Another component of the motion score is derived from optical flow information. We use the Recurrent All-Pairs Field Transforms (RAFT) neural network [38] to generate dense optical flow. To extract motion features, we consider the region of interest defined by an object's bounding box. This region is divided into a 3×3 grid, and for each grid cell, we compute the average magnitude and orientation of the optical flow. These values are used to determine a motion vector for each cell. Each grid cell casts a vote for the most likely position of the object in the next frame. Specifically, we use the detection from the previous frame that was previously associated with a track (and used to update it). The motion score is then computed based on the total number of votes linking this past detection to a new detection in the current frame, helping to determine the best match for updating the track. The appearance score includes multiple elements. First, it considers similarity costs related to the geometric properties of the compared detection and track. These similarity scores are computed using the L1 norm (1).

$$L_1(x, y) = \sum |x_i - y_i|_1 \quad (1)$$

The geometric properties used in this calculation include object width, height and area. Additionally, the similarity score includes the Intersection over Union (IoU) score between the track and the detection.

Besides geometric properties, the proposed approach also leverages a Siamese network, which produces a similarity score between the two objects compared. This network has an architecture facilitating the high running time of the object association and includes three main submodules, each consisting of: ReflectionPad2d, Conv2D, LeakyRelu and BatchNorm. These layers process the input tensors and extract relevant features. After these submodules, fully connected layers with LeakyReLU activation functions further aggregate the features and generate high-level representations. The embeddings resulted from the two inputs are then compared using the Euclidean distance to determine the degree of similarity. The network has been trained using contrastive learning using the dataset presented in section III D. All elements in the similarity score are weighted using empirically determined weights based on experiments. The final score is obtained by summing the motion similarity score and the appearance similarity score (2).

$$S_{final} = S_{motion} + S_{appearance} \quad (2)$$

After computing the similarity scores, the Hungarian algorithm [24] is used to optimally assign detections to tracks. Each tracked object maintains an embedded histogram with six bins corresponding to different semantic classes: person, car, rider, bicycle, animal, and traffic element. Whenever a detection is associated with a track, the semantic class of that detection casts a vote in the corresponding bin of the histogram. Over time, as more associations occur, the histogram accumulates votes reflecting the most frequently observed class for that tracked object. The semantic class of the track is determined by selecting the class with the highest vote count in the histogram. In case of ties, the class corresponding to the first bin (in a predefined order) is chosen. The tracking stages related to track management (birth and deletion of tracks) remain unchanged, following the approach presented in [23]. The trajectory of tracked objects is filtered using a Kalman Filter, and in each update step, all available detection information including object geometrical properties, instance mask, and other attributes are incorporated into the tracked object.

Figure 2. Tracking multiple objects across frames

In Figure 2 we can observe the result of the tracking module, where objects up to 100m (in depth) 30m (latera) are tracked. It can be observed that the unique ID assigned to the car and the truck and is color coded with a unique color is maintained across frames. After this step the tracking module will send the tracked objects with all their properties including instance masks and bounding boxes to the BEV Projection and Filtering Module.

C. BEV transformation

After obtaining the disparity image from the monocular depth estimation module, we compute the scaled 3D points using the approach in [39], to recover the absolute depth information. We then create an enhanced point cloud by storing for each 3D point the intensity information and semantic label. This can be easily achieved, because for each disparity point we know the corresponding pixel in the intensity and semantic segmentation images. In the process of projecting a 3D point cloud onto a Bird's Eye View (BEV) image, the x and z coordinates must be transformed to fit within a fixed image size. First, the bounding limits of the point cloud are determined, defining the minimum and maximum values for x and z, denoted as $x_{min}, x_{max}, z_{min}$ and z_{max}. To map these coordinates into the image space, a scaling factor is computed for x and z dimensions and to maintain the aspect ratio and ensure the entire point cloud fits within the image, the smallest of these scaling factors is selected. To horizontally center the projection in the image, an x offset is introduced, computed as in (3).

$$offset_x = (width - (x_{min} - x_{max}) * scale) \,/\, 2 \qquad (3)$$

For the z coordinate, the goal is to align the bottom of the image with z_{max} and the top with z_{min}. Therefore, the z offset is computed as in (4)

$$offset_z = (height - (z_{min} - z_{max}) * scale) \qquad (4)$$

The variables height and width represent the height and width of the BEV image. Once the scaling and offsets are determined, each 3D point is projected onto the BEV image using the expression in (5) and (6)

$$u = (x - x_{min})\, scale + offset_x \qquad (5)$$
$$v = (z - z_{min})\, scale + offset_z \qquad (6)$$

where (u, v) represent the 2D pixel coordinates in the BEV image. For each (u, v) we place the color corresponding to the "stuff" semantic class in the BEV image from the enhanced point cloud. For projecting the object instances, we create a 4D occupancy grid having cells of 35cm x 70cm. One dimension from this 4D occupancy grid is storing the cell density (how many 3D points fall within that cell), one dimension is storing minimum distance from that cell to the ego vehicle, and 2 of the dimensions are storing the real-world X and Z values corresponding to the point for which the minimum distance was obtained. As we iterate through the depth values corresponding to the instance masks, each depth value casts a vote in the cell of the grid where it falls. We transform the density cells into probabilities and then perform an adaptive thresholding of the grid, leaving only the cells for which the density is higher than a probability threshold. Since the cells that are closer to the ego vehicle are expected to be denser, then the cells that are further away the probability threshold for the closer cells is 0.92 and this gradually decays to the end of the grid where for a cell to remain active there should be 0.4 probability. Since projected objects in BEV can become fragmented due to artifacts that appear when transforming an intensity image—where points may spread laterally and fill unintended grid cells—we apply a clustering step. After the adaptive thresholding, we label the remaining clusters and retain only the largest one, effectively filtering out noise cells. The labeling is performed using the two-pass labeling method with equivalence classes, and the largest cluster is determined by counting the pixels in each cluster. In the next step, we compute the closest point of the instance using the available cells of the grid and eliminate all the points that are at a distance larger than 6m to that point. The three steps mentioned above can be seen in Figure 3. In the left-hand side, there are the original projected points belonging to an object instance, in the middle we have the largest cluster, and finally in the right-hand side we display the filtered largest cluster.

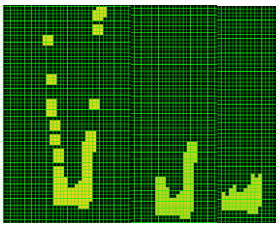

Figure 3. The first three steps of the grid filtering algorithm

The facets of the filtered 2D grid objects are identified, and for each facet segment, the shortest distance to its closest point from the ego-vehicle's position is computed. Considering we want to find the distance from a reference point (x_0, y_0) to a segment of coordinates $[(x_1, y_1), (x_2, y_2)]$ the parametric projection of the point onto the line segment is given by (7).

$$t = \frac{(x_0 - x_1)(x_2 - x_1) + (y_0 - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2} \qquad (7)$$

For t < 0 the closest point on the segment is (x_1, y_1); conversely, if t > 1 the closest point is (x_2, y_2). For values within the interval [0, 1], the projection falls within the segment and is computed using (8) and (9).

$$x_{closest} = x_1 + t(x_2 - x_1) \qquad (8)$$
$$y_{closest} = y_1 + t(y_2 - y_1) \qquad (9)$$

Facets are sorted by their Euclidean distance to the ego vehicle, with the most visible ones being closest. The intersection point of the two most visible facets is determined and stored as a reference. Furthermore, the longest of these two visible facets is used to determine the object's orientation angle and whether it is observed from the lateral, front, or rear view. Using the endpoints $[(x_1, y_1), (x_2, y_2)]$, of the longest facet we compute the orientation angle θ of this facet in degrees (10).

$$\theta = \frac{\operatorname{atan}(y_2 - y_1, x_2 - x_1) * 180}{\pi} \qquad (10)$$

Using this angle, we define an interval to determine whether the object is observed from the front, rear, or lateral view. Furthermore, given the reference point, the object's orientation angle, its semantic class, and its dimensions, we reconstruct the corresponding rectangle in the BEV space. In Figure 4 we can

observe a scene with validated objects (they have blue bounding boxes) and their projection in the BEV space

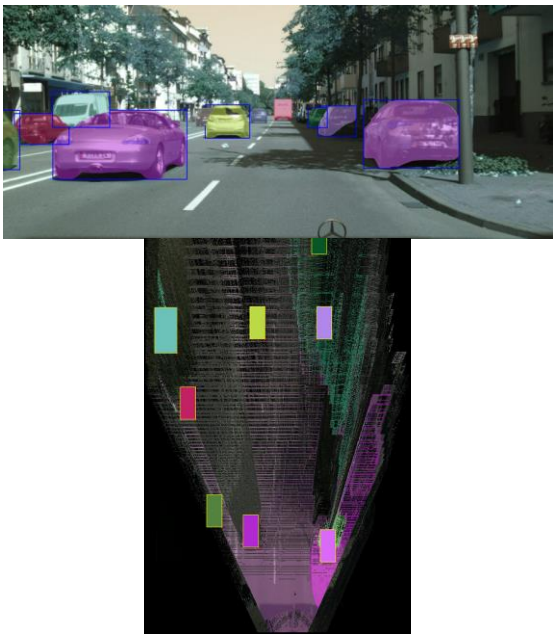

Figure 4. BEV representation in multiple objects scenario

It is important to note that the current approach is working even in difficult situations. For example, in situations where multiple instance masks of objects intersect, the depth estimation module may generate erroneous points especially at object boundaries, making some objects appear closer than they are or even overlapping with other instances. However, due to the proposed BEV filtering approach such situations are eliminated in the proposed solution. Such a scenario is presented in Figure 5. An extra inpainting post processing step can be implemented if the dark regions in the BEV map are considered undesirable.

D. Dataset used for similarity learning

Siamese networks generate feature embeddings from data pairs, using an energy function to evaluate input similarity. To train such a network for distinguishing object types, a dataset was constructed with four categories: cars and pedestrians (1,000 instances each), and riders and animals (500 instances each). Each instance includes at least 10 images, totaling over 30,000 samples captured from various angles and distances. The dataset spans diverse conditions (day/night, rain, fog) and modalities (RGB, grayscale, thermal). Data sources include public datasets [28–33], YouTube, and sequences recorded by us, following ethical guidelines. Source distribution is 30% public datasets, 20% YouTube, and 50% our own recordings.

For creating the dataset, we have considered the following criteria:

- The relevance of the images sequence for the classes of objects of interest. Only traffic sequences containing the four types of objects of interest were selected.
- Image quality – frames that had a good quality with a high resolution, a good contrast and reduced noise were preferred.
- Image diversity – the sequences that captured objects of interest from many different vantage points were preferred.
- Variations of lightning conditions. Images where the same instance was captured in different lightning conditions, or images in different environments (urban, rural, indoor, outdoor) were preferred.

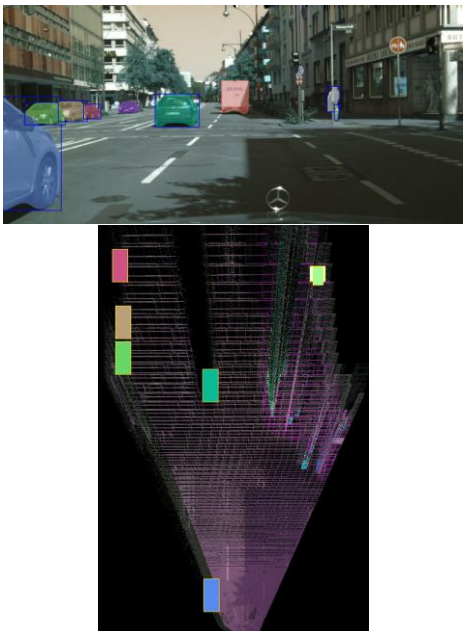

Figure 5. Scenario where the instance masks intersect

The dataset was created in three phases. First, object instances of interest were manually cropped from each frame and saved into folders named after their respective sequences. In the second phase, images of the same object instance were grouped into individual folders, which were then organized under their corresponding object class. A final cross-check ensured the removal of duplicates. For datasets like [31], where instances were already pre-cropped, the task was limited to organizing them by viewpoint diversity within each class. The final dataset structure includes a main 'Dataset' folder with four subfolders (one per class), each containing folders for individual instances. Sample images are shown in Figure 6. The dataset is available at: https://users.utcluj.ro/~mmp/IV2025/ .

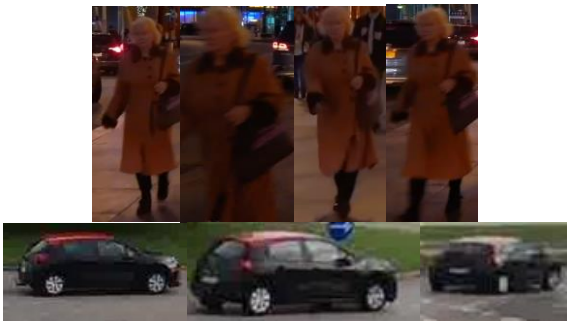

Figure 6. Example of images from the proposed dataset

IV. Experimental Results

The approach was implemented in C++ using neural network models trained in PyTorch and converted to LibTorch for C++ compatibility. ONNX and TensorRT were used with the Depth Anything v2 module. OpenCV handled visualization

and image merging, while the Point Cloud Library managed 3D data. Implementation ran on an Intel Core i7-11370H (3.3 GHz) and an Nvidia RTX 3070 GPU. Training data was split into 70% training, 10% validation, and 20% testing. Evaluation on the KITTI dataset (car category) used HOTA, MOTA, and AssA metrics. The numerical results are presented in Table I.

TABLE I. EVALUATION OF THE TRACKING APPROACH ON KITTI

Method	*HOTA*	*MOTA*	*ASSA*
Proposed Approach	**79.21**	**89.8**	**82.29**
UG3DMOT [40]	78.60	87.98	82.28
MSA-MOT [41]	78.52	88.01	82.56
YONTD-MOT[42]	78.08	85.09	82.86
PermaTrack[43]	78.03	91.33	78.41

It is important to note that the performance of the EfficientPS model does not suffer any significant changes after the modifications made to its architecture needed for it to be exported in C++.

The performance of the BEV detector was evaluated both quantitatively using standard metrics such as Accuracy, Precision, Recall, and visually to observe obvious errors and validate the consistency of predictions in selected examples. Additionally, we introduce a custom metric Δ to evaluate object placement correctness in BEV space, as standard metrics such as IoU or mAP do not fully capture projection accuracy in monocular BEV reconstruction tasks. From the total number of objects detected from the perspective view, we verify the number of objects correctly placed. The metric for frame i is presented in equation (11).

$$\Delta_i = \frac{number\ of\ object\ correctly\ placed}{total\ number\ of\ objects} \qquad (11)$$

To assess localization accuracy, the number of correctly placed objects is determined by subtracting, from the total number of detections, the objects that either do not appear in the BEV representation (but are present in the perspective view) or are incorrectly placed. The mean placement error Δ is computed as the average of the individual frame errors Δ_k. This evaluation was carried out on the Cityscapes dataset. Furthermore, we compared our approach against several fully data-driven deep learning methods trained on the KITTI 360 dataset for BEV scene reconstruction, using the proposed evaluation metric. The results are presented in Table II. Here, Accuracy (A) denotes the ratio of correctly localized objects to the total number in the BEV view, Precision (P) measures the ratio of correctly matched objects to total BEV detections, and Recall (R) quantifies the correctly matched objects relative to the total number in the perspective view. The analysis was limited to objects classified as cars.

The proposed solution is the best-performing, with the highest accuracy, precision, and recall from the compared methods. The situation in which the proposed solution does not correctly place objects is when the object instance is heavily occluded, and there is a small part of it still visible and the 3D points on that small part are not reliable.

TABLE II. EVALUATION OF COMBINED MODELS ON CITISCAPES

Method	Δ	A	P	R
YoloV8n + DeepLabV3 Plus Bev	0.40	0.52	0.55	0.45
Modified Efficient PS BEV	0.63	0.73	0.77	0.68
YoloV8x-Seg + DeepLabV3 Plus BEV	0.74	0.80	0.81	0.75
Proposed Solution	**0.88**	**0.92**	**0.90**	**0.91**

The running time of the proposed pipeline on images from the Cityscapes dataset is approximately 250ms. To reduce complexity, a shared backbone for monocular depth estimation, panoptic segmentation, and optical flow could be employed. However, due to computational and storage limitations, the current approach utilizes pretrained models for Depth Anything v2 and optical flow, leading to a non-multitask architecture. Additionally, during validation, the redundant object detector must operate with a distinct feature extractor, separate from the main network for obtaining the best results.

V. CONCLUSIONS

In this paper we have presented an original framework that performs multi-object tracking and panoptic segmentation in Bird's Eye View space by integrating deep learning methods, grid filtering, and feature engineering techniques. The proposed solution significantly enhances object tracking accuracy through a hybrid data association function that combines engineered features with Siamese neural networks. We also presented a modified version of EfficientPS that can be exported and integrated in C++. The validation approach ensures that the environment is correctly represented by eliminating unwanted detections or misclassified objects. The dataset created for training the Siamese network from the data association function includes images from four object categories and serves as a valuable resource for future research and applications. The proposed solution was evaluated on sequences from international publicly available benchmarks.

ACKNOWLEDGMENT

This work is supported by the project "Romanian Hub for Artificial Intelligence-HRIA", Smart Growth, Digitization and Financial Instruments Program, MySMIS no. 334906

REFERENCES

[1] J. V. Hurtado, R. Mohan, W. Burgard, and A. Valada, "Mopt: Multi-object panoptic tracking," arXiv preprint arXiv:2004.08189, 2020

[2] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in European Conf. on Computer Vision, 2020

[3] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in IEEE Conf. on Computer Vision and Pattern Recognition, June 2020

[4] N. Radwan, W. Burgard, and A. Valada, "Multimodal interaction-aware motion prediction for autonomous street crossing," The International Journal of Robotics Research, vol. 39, no. 13, pp. 1567–1598, 2020

[5] N. Gosala and A. Valada, "Bird's-Eye-View Panoptic Segmentation Using Monocular Frontal View Images," in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1968-1975, April 2022.

[6] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chong-hao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In European conference on computer vision, pages 1–18. Springer, 2022

[7] L. Reiher, B. Lampe, and L. Eckstein, "A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view," in Int. Conf. on Intelligent Transportation Systems, 2020.

[8] R. Danescu, R. Itu and M. P. Muresan, "PartID – Individual Objects Tracking in Occupancy Grids Using Particle Identities," *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, Romania, 2020, pp. 283-290

[9] M. H. Ng, K. Radia, J. Chen, D. Wang, I. Gog, and J. E. Gonzalez, "Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud," arXiv preprint arXiv:2006.11436, 2020

[10] Q. Liu, X. Li, Z. He, N. Fan, D. Yuan and H. Wang, "Learning DeepMulti-Level Similarity for Thermal Infrared Object Tracking," in IEEETransactions on Multimedia, vol. 23, pp. 2114-2126, 2021, doi:10.1109/TMM.2020.3008028.

[11] H. Karunasekera, H. Wang and H. Zhang, "Multiple Object TrackingWith Attention to Appearance, Structure, Motion and Size," in IEEEAccess, vol. 7, pp. 104423-104434, 2019, doi:10.1109/ACCESS.2019.2932301.

[12] M. Betke and Z. Wu, *Data Association for Multi-Object Visual Tracking*. 1st ed. Cham: Springer, 2017.

[13] J. Philion and S. Fidler, "Lift, splat, scityhoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV*, vol. 16, pp. 194–210, Springer, 2020

[14] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN: Towards high-performance voxel-based 3D object detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 2, pp. 1201–1209, 2021

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12697–12705, 2019

[16] V. A. Sindagi, Y. Zhou, and O. Tuzel, "MVX-Net: Multimodal VoxelNet for 3D object detection," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7276–7282, 2019.

[17] T. Huang, Z. Liu, X. Chen, and X. Bai, "EPNet: Enhancing point features with image semantics for 3D object detection," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV*, vol. 16, pp. 35–52, Springer, 2020.

[18] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In European conference on computer vision, pages 1–18. Springer, 2022

[19] owen Pan, Jiankai Sun, Alex Andonian, Aude Oliva, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. CoRR, abs/1906.03560, 2019

[20] N. Gosala and A. Valada, "Bird's-Eye-View Panoptic Segmentation Using Monocular Frontal View Images," in *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1968-1975, April 2022

[21] H. Karunasekera, H. Wang and H. Zhang, "Multiple Object TrackingWith Attention to Appearance, Structure, Motion and Size,"in IEEEAccess, vol. 7, pp. 104423-104434, 2019, doi:10.1109/ACCESS.2019.2932301

[22] P. Voigtlaender *et al.*, "MOTS: Multi-Object Tracking and Segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 7934-7943

[23] M. P. Muresan, R. Danescu and S. Nedevschi, "Multi-Object Tracking, Segmentation and Validation in Thermal Images," *2023 IEEE Intelligent Vehicles Symposium (IV)*, Anchorage, AK, USA, 2023, pp. 1-8

[24] Kuhn, H.W. The Hungarian method for the assignment problem. Nav.Res. Logist. Q. 1955, 2, 83–97

[25] K. He, G. Gkioxari, P. Dollar, and R. Girshick, ''Mask R-CNN,'' inProc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2961–2969

[26] M. P. Muresan, S. Nedevschi, and R. Danescu, "Robust DataAssociation Using Fusion of Data-Driven and Engineered Features forReal-Time Pedestrian Tracking in Thermal Images," Sensors, vol. 21,no. 23, p. 8005, Nov. 2021

[27] Y. Yang, Y. Deng, J. Fan, J. Zhang, and Z.-J. Zha, "BEVTrack: A Simple and Strong Baseline for 3D Single Object Tracking in Bird's-Eye View," *arXiv e-prints*, p. arXiv-2309, 2024.

[28] L. Ma, H. Liu, L. Hu, C. Wang și Q. Sun, „Orientation Driven Bag of Appearances for Person Re-identification," CoRR, vol. abs/1605.02464, 2016

[29] G. Song, B. Leng, C. Hetang și S. Cai, „Region-Based Quality Estimation Network for Large-Scale Person Re-Identification," în Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), NewOrleans, LA, USA, 2018

[30] X. Liu, W. Liu, T. Mei și H. Ma, „Progressive and Multimodal Vehicle Reidentification for," IEEE Transactions on Multimedia, vol. 20, nr. 3, pp. 645- 658, 2018

[31] A. Kumar, M. H. Takehiro Kashiyama, F. Zhang, H. Omata și Y. Sekimoto„Vehicle re-identification and trajectory reconstruction using multiple moving cameras in the CARLA driving simulator," în BigData Conference, Osaka, Japan, 2022

[32] „FREE Teledyne FLIR Thermal Dataset for Algorithm Training," Teledyne

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[34] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[35] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[36] R. Mohan și A. Valada, *"EfficientPS: Efficient Panoptic Segmentation," Int. J. Comput. Vis.*, vol. 129, pp. 1551–1579, 2020

[37] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, și H. Zhao, *"Depth Anything V2,"* arXiv preprint arXiv:2406.09414, 2024

[38] Z. Teed și J. Deng, *"RAFT: Recurrent All-Pairs Field Transforms for Optical Flow,"* în *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020

[39] F. Xue, G. Zhuo, Z. Huang, W. Fu, Z. Wu and M. H. Ang, "Toward Hierarchical Self-Supervised Monocular Absolute Depth Estimation for Autonomous Driving Applications," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2020, pp. 2330-2337

[40] Jiawei He, Chunyun Fu, Xiyang Wang, and Jianwen Wang, "3D multi-object tracking based on informatic divergence-guided data association," *Signal Processing*, vol. 222, p. 109544, 2024.

[41] Z. Zhu, J. Nie, H. Wu, Z. He, and M. Gao, "MSA-MOT: Multi-Stage Association for 3D Multimodality Multi-Object Tracking," *Sensors*, vol. 22, no. 22, p. 8650, 2022.

[42] X. Wang, C. Fu, J. He, M. Huang, T. Meng, S. Zhang, H. Zhou, Z. Xu, and C. Zhang, "You Only Need Two Detectors to Achieve Multi-Modal 3D Multi-Object Tracking," *arXiv preprint*, arXiv:2304.08709, 2024

[43] P. Tokmakov, J. Li, W. Burgard, și A. Gaidon, *"Learning To Track With Object Permanence,"* în *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, oct. 2021, pp. 10860–10869.