# Efficient and Robust 2D-to-BEV Representation Learning via Geometry-guided Kernel Transformer

Shaoyu Chen*, Tianheng Cheng*, Xinggang Wang†, Wenming Meng, Qian Zhang, Wenyu Liu



**Abstract**—Learning Bird's Eye View (BEV) representation from surrounding-view cameras is of great importance for autonomous driving. In this work, we propose a Geometry-guided Kernel Transformer (GKT), a novel 2D-to-BEV representation learning mechanism. GKT leverages the geometric priors to guide the transformer to focus on discriminative regions, and unfolds kernel features to generate BEV representation. For fast inference, we further introduce a look-up table (LUT) indexing method to get rid of the camera's calibrated parameters at runtime. GKT can run at 72.3 FPS on 3090 GPU / 45.6 FPS on 2080ti GPU and is robust to the camera deviation and the predefined BEV height. And GKT achieves the state-of-the-art real-time segmentation results, i.e., 38.0 mIoU (100m×100m perception range at a 0.5m resolution) on the nuScenes val set. Given the efficiency, effectiveness and robustness, GKT has great practical values in autopilot scenarios, especially for real-time running systems. Code and models will be available at https://github.com/hustvl/GKT.

**Index Terms**—Autonomous driving, 3D perception, Bird's Eye View, Robustness, LUT Indexing.

## 1 Introduction

Surrounding-view perception based on Bird's Eye View (BEV) representation is a cutting-edge paradigm in autonomous driving. For multi-view camera system, how to transform 2D image representation to BEV representation is a challenging problem. According to whether geometric information is explicitly leveraged for feature transformation, previous methods can be divided into two categories, *i.e.*, geometry-based pointwise transformation and geometry-free global transformation.

**Geometry-based Pointwise Transformation** As illustrated in Fig. 1(a), pointwise transformation methods [1], [2], [3], [4], [5], [6] leverage camera's calibrated parameters (instrinsics and extrinsics) to determine the correspondence (one-to-one or one-to-many) between 2D positions and BEV grids. With the correspondence available, 2D features are projected to 3D space and form BEV representation.

However, pointwise transformation relies too much on the calibrated parameters. For a running autopilot system, because of the complicated external environment, cameras might deviate from the calibrated position at runtime, which makes the 2D-to-BEV correspondence unstable and brings system error to the BEV representation. Besides, pointwise transformation usually requires complicated and time-consuming 2D-3D mapping operations, *e.g.*, predicting depth probability distribution over pixels, broadcasting pixel features along the ray to BEV space, and high precision calculation about camera's parameters. These operations are inefficient and hard to be optimized, limiting the real-time applications.

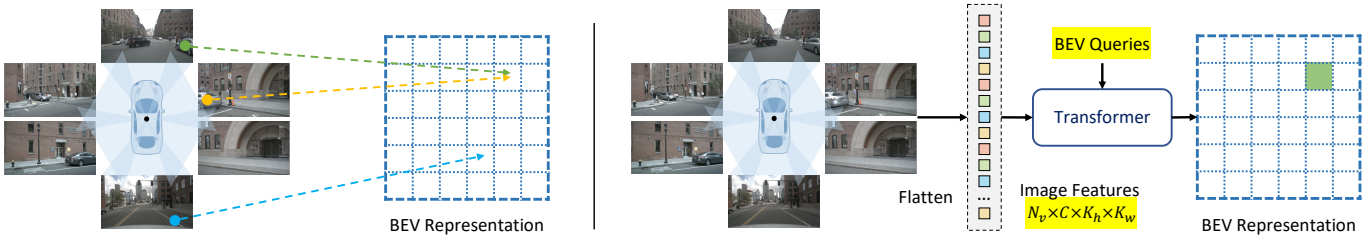
**Geometry-free Global Transformation** Global transformation methods [7] consider the full correlation between image and BEV. As shown in Fig. 1(b), multi-view image features are flattened and each BEV grid interacts with all image pixels. Global transformation does not rely on the geometric priors in 2D-to-BEV projection. Thus, it's insensitive to the camera deviation.

But it also raise problems. 1) The computational budget of global transformation is proportional to the number of image pixels. There exists sharp contradiction between resolution and efficiency. 2) Without geometric priors as guidance, the model has to globally dig out discriminative information from all views, which makes convergence harder.

In this work, targeting at efficient and robust BEV representation learning, we propose a new 2D-to-BEV transformation mechanism, named Geometry-guided Kernel Transformer (GKT for short). With coarse cameras' parameters, we roughly project BEV positions to get prior 2D positions in multi-view and multi-scale feature maps. Then, we unfold $K_h \times K_w$ kernel features around the prior positions, and make BEV queries interact with corresponding unfolded features to generate BEV representation. Further, we introduce LUT indexing to get rid of camera's parameters at runtime.

GKT is of high robustness at runtime. Compared with pointwise transformation, GKT only takes camera's parameters as guidance but not rely too much on them. When camera deviates, correspondingly the kernel regions shift but can still cover the targets. Transformer is permutation-invariant and the attention weights for the kernel regions are dynamically generated according to the deviation. Thus, GKT can always focus on the targets, insensitive to the

- *This work is still in progress.*
- *Shaoyu Chen, Tianheng Cheng, Xinggang Wang and Wenyu Liu are with Huazhong University of Science and Technology. Wenming Meng and Qian Zhang are with Horizon Robotics. This work is done when Shaoyu Chen and Tianheng Cheng are interns at Horizon Robotics.*
- * *Equal contribution.*
- † *Corresponding author.*




(a) Geometry-based Pointwise Transformation

(b) Geometry-free Global Transformation

Fig. 1: (a) Geometry-based pointwise transformation leverages camera's calibrated parameters (instrinsics and extrinsics) to determine the correspondence (one to one or one to many) between 2D positions and BEV grids. (b) Geometry-free global transformation considers the full correlation between image and BEV. Each BEV grid interacts with all image pixels.

deviation of camera.

GKT is of high efficiency. With the proposed LUT indexing, at runtime we get rid of 2D-3D mapping operations required by pointwise transformation, making the forward process compact and fast. And compared with global transformation, GKT only focuses on geometry-guided kernel regions, avoiding global interaction. GKT requires less computation and convergences faster.

Consequently, GKT well balances between pointwise and global transformation, leading to efficient and robust 2D-to-BEV representation learning. We validate GKT on nuScenes map-view segmentation. GKT is promisingly efficient, running at **72.3** FPS on 3090 GPU / **45.6** FPS on 2080ti GPU, much faster than all existing methods. And GKT achieves **38.0** mIoU, which is SOTA among all real-time methods. We will extend GKT to other BEV-based tasks in the near future.

## 2 Method

### 2.1 Geometry-guided Kernel Transformer

The framework of the proposed GKT is presented in Fig 2. Shared CNN backbone extracts multi-scale multi-view features $\mathcal{F}_{\text{img}} = \{F_v^s\}$ from surround-view images $\mathcal{I} = \{I_v\}$. The BEV space is evenly divided into grids. Each BEV grid corresponds to a 3D coordinate $P_i = (x_i, y_i, z)$ and a learnable query embedding $q_i$. $z$ is the predefined height of BEV plane shared by all queries. GKT is insensitive to the value of $z$, which is further discussed in Sec. 3.5.

We leverage the geometric priors to guide the transformer to focus on discriminative regions. With camera's parameters, we roughly project each BEV grid $P_i$ to a set of float 2D pixel coordinates $\{Q_i^{sv}\}$ (for different views and scales) and then round them to integer coordinates $\{\bar{Q}_i^{sv}\}$, *i.e.*,

$$\begin{aligned} Q_i^{sv} &= \mathbf{K}^{sv} \cdot \mathbf{Rt}^{sv} \cdot P_i^{sv}, \\ \bar{Q}_i^{sv} &= round(Q_i^{sv}) \end{aligned} \tag{1}$$

We unfold $K_h \times K_w$ kernel regions around the prior positions $\{\bar{Q}_i^{sv}\}$. It's worth noting that if the kernel regions exceed the image boundary the exceeding part is set to zero. Each BEV query $q_i$ interacts with the corresponding unfolded kernel features $F \in \mathbb{R}^{N_{view} \times N_{scale} \times C \times K_h \times K_w}$ and generates BEV presentation. Heads for various tasks (*e.g.*, detection, segmentation, motion planning) can be performed on the BEV presentation.

### 2.2 Robustness to Camera Deviation

For a running autopilot system, the external environment is complicated. Camera will deviate from its calibrated position. GKT is robust to the camera deviation. To validate this, we simulate the deviation of camera in real scenarios. Specifically, we decompose the deviation to rotation deviation $\mathbf{R}_{devi}$ and translation deviation $\mathbf{T}_{devi}$ and add random noise to all the $x, y, z$ dimensions.

The translation deviation $\mathbf{R}_{devi}$ is formulated as,

$$\mathbf{T}_{devi} = \begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

The rotation deviation $\mathbf{R}_{devi}$ is formulated as,

$$\mathbf{R}_{devi} = R_{\theta_x} \cdot R_{\theta_y} \cdot R_{\theta_z} \tag{3}$$

$$\begin{aligned} R_{\theta_x} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & sin(\theta_x) & 0 \\ 0 & -sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ R_{\theta_y} &= \begin{bmatrix} cos(\theta_y) & 0 & -sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\theta_y) & 0 & cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ R_{\theta_z} &= \begin{bmatrix} cos(\theta_z) & sin(\theta_z) & 0 & 0 \\ -sin(\theta_z) & cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4}$$

$\Delta x, \Delta y, \Delta z, \theta_x, \theta_y, \theta_z$ are all random variables subject to normal distribution. *I.e.*,

$$\begin{aligned} \Delta x, \Delta y, \Delta z &\sim \mathcal{N}(0, {\sigma_2}^2) \\ \theta_x, \theta_y, \theta_z &\sim \mathcal{N}(0, {\sigma_1}^2) \end{aligned} \tag{5}$$

$\theta_x, \theta_y, \theta_z$ correspond to the rotation noise and $\Delta x, \Delta y, \Delta z$ correspond to the translation noise, respectively relative to the the $x, y, z$ axes of camera coordinate system.

We add random noise to all the $x, y, z$ dimensions and all cameras. With deviation noise introduced, Eq. 1 becomes

$$\begin{aligned} Q_i^{sv} &= \mathbf{K}^{sv} \cdot \mathbf{R}_{devi} \cdot \mathbf{T}_{devi} \cdot \mathbf{Rt}^{sv} \cdot P_i^{sv}, \\ \bar{Q}_i^{sv} &= round(Q_i^{sv}) \end{aligned} \tag{6}$$

Experiments about the camera deviation are presented in Sec. 3.4. GKT is robust to the deviation. When camera

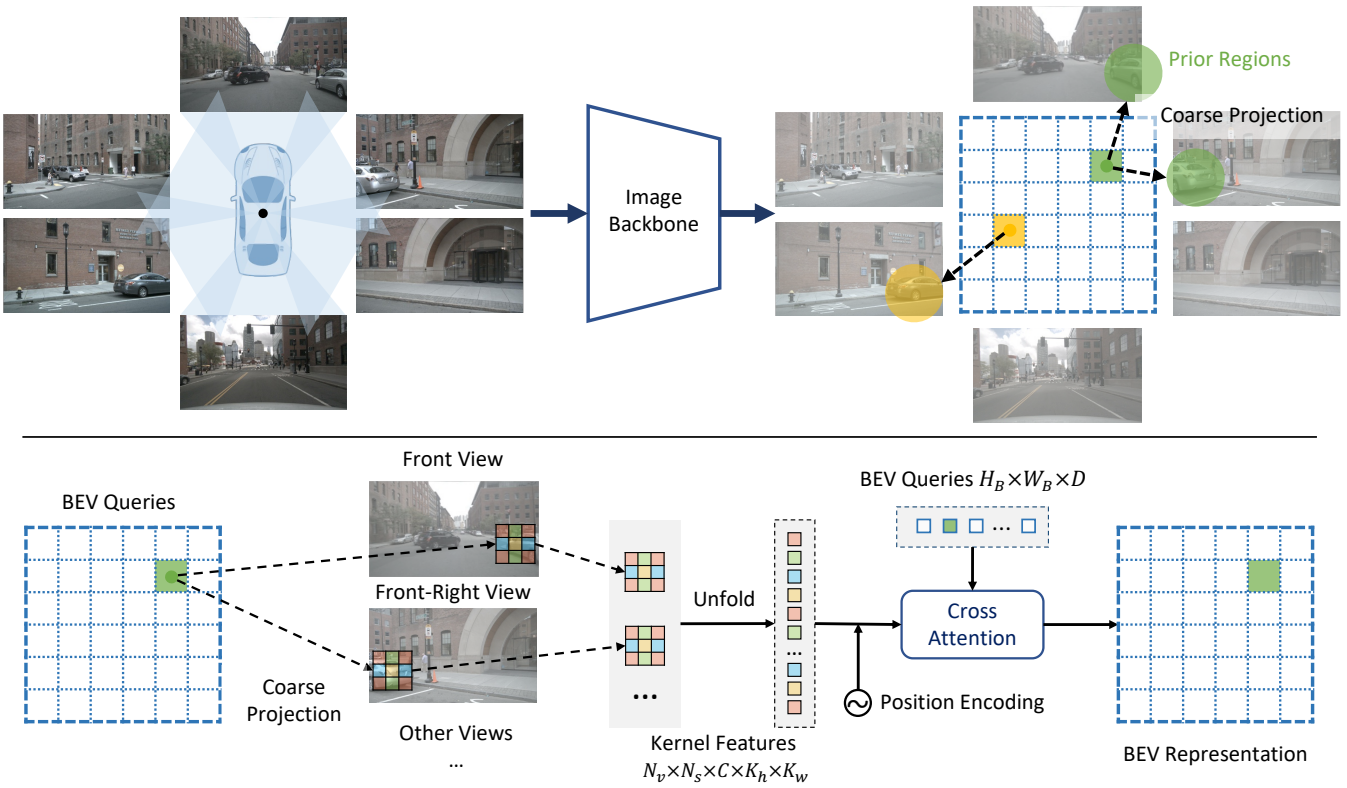



Fig. 2: Illustration of GKT. Top: geometric information is leveraged to guide the transformer to focus on prior regions in multi-view images. Bottom: we unfold kernel features of the prior regions and make them interact with BEV queries to generate BEV representation.

deviates, the prior regions shift but can still cover the targets. And the rounding operation in Eq 1 is anti-noise. When the camera's parameters slightly change, after rounding the coordinates keep the same. Besides, transformer is permutation-invariant and the attention weights for the kernel regions are dynamic according to the deviation.

### 2.3 BEV-to-2D LUT Indexing

We further introduce BEV-to-2D LUT Indexing to speed up GKT. The kernel regions for each BEV grid is fixed and can be pre-computed offline. Before runtime, we construct a LUT (look-up table) which caches the correspondence between indices of BEV queries and indices of image pixels. At runtime, we get corresponding pixel indices for each BEV query from the LUT, and efficiently fetch kernel features through indexing.

With LUT Indexing, GKT gets rid of high precision calculation about camera's parameters and achieves higher FPS. In Sec. 3.7, we compare other implementation manners with LUT Indexing to validate the efficiency.

### 2.4 Configuration of Kernel

For GKT, the configuration of kernel is flexible. We can adjust the kernel size to balance between the receptive field and computation cost. And the layout of kernel is feasible (cross shape kernel, dilated kernel, *etc.*). Since LUT indexing is adopted to fetch kernel features, the efficiency of GKT is not influenced by the layout of kernel.

## 3 Experiments

In this section, we mainly evaluate the proposed GKT on nuScenes map-view segmentation and conduct extensive experiments to investigate GKT, including the convergence speed and robustness to the camera deviation.

### 3.1 Dataset

The nuScenes dataset [8] contains 1,000 driving sequences with 700, 150, and 150 for training, validation, and testing, respectively. Each sequence lasts 20 nearly seconds and contains 6 surrounding-view images around the ego-vehicle per frame. We resize the 6-camera images into $224 \times 480$ for both training and validation. All models are trained on nuScenes *train* set and evaluated on the *val* set.

### 3.2 Implementation Details

We adopt CVT [7] as the basic implementation for BEV map-view segmentation. Following [7], we feed the images into an EfficientNet-B4 [9] to extract multi-scale image features. We randomly initialize the $25 \times 25$ BEV feature maps as queries and adopt the proposed GKT to transform image features to BEV features. Then three convolutional blocks with upsampling layers are employed to upsample BEV features to $200 \times 200$ for map-view segmentation.

All models are trained on 4 NVIDIA GPUs with 4 samples per GPU. We adopt the training schedule from [7] and train the GKT with 30k iterations and AdamW optimizer, which takes nearly 3 hours for training. Following [7], we

| Method | Setting 1 | Setting 2 | FPS | Params |
|---|---|---|---|---|
| VPN [10] | 25.5 | - | - | - |
| STA [11] | 36.0 | - | - | - |
| FIERY [1] | 37.7 | 35.8 | - | - |
| FIERY† [1] | 42.7 | 39.8 | 8.0 | 7.4M |
| BEVFormer [3] | - | 43.2 | 1.7* | 68.1M |
| PON [5] | 24.7 | - | 30.0 | - |
| Lift-Splat [2] | - | 32.1 | 25.0 | 14.0M |
| CVT [7] | 37.2 | 36.0 | 35.0 | 1.1M |
| CVT† [7] | 39.3 | 37.2 | 34.1 | 1.1M |
| GKT | 41.4 | 38.0 | 45.6 | 1.2M |

TABLE 1: **Vehicle map-view segmentation on nuScenes.** FPS is measured on 2080Ti GPU, except * denotes measured on V100. † denotes our reproduced results, which are higher than the ones reported in the original paper.

adopt two evaluation settings, *i.e.*, **setting 1** and **setting 2**, which evaluate the segmentation results with 100m×50m perception range at a 0.25m resolution and 100m×100m perception range at a 0.5m resolution, respectively. Unless specified, we adopt **setting 2** as the default evaluation setting.

### 3.3 Main Results

In Tab. 1, we compare GKT with other BEV-based methods on vehicle map-view segmentation with two evaluation settings. Specifically, we adopt a $1 \times 7$ convolution for capturing the horizontal context and then apply an efficient $7 \times 1$ kernel for 2D-to-BEV transformation in GKT. FIERY [1] and BEVFormer [3] achieve high performance but are time-consuming, far away from real-time applications. GKT can run at $45.6$ FPS on 2080Ti with $41.4$ mIoU for **setting 1** $38.0$ mIoU for **setting 2**, achieving the highest efficiency and performance among all existing real-time methods.

### 3.4 Robustness to Camera Deviation

To validate the robustness, we traverse the validation sets (with $6019$ samples) and randomly generate noise for each sample. We respectively add translation and rotation deviation of different degrees. Note that we add noise to all cameras and all coordinates. And the noise is subject to normal distribution. There exists extremely large deviation in some samples, which affect the performance a lot. As shown in Tab. 2 and Tab. 3, when the standard deviation of $\Delta_x, \Delta_y, \Delta_z$ is $0.5m$ or the standard deviation of $\theta_x, \theta_y, \theta_z$ is $0.02rad$, GKT still keeps comparable performance.

We observe that $5 \times 5$ kernel is more robust to the deviation than $3 \times 3$ kernel. It proves larger kernel size corresponds to stronger robustness. And $7 \times 3$ kernel is more robust to the deviation than $5 \times 5$ kernel. 7 ($k_h$) corresponds to the vertical dimension of the 3D space. For each BEV grid, $x, y$ is fixed but $z$ is predefined. There exists more uncertainty in vertical dimension. Thus, adopting to a larger $k_h$ is better.

### 3.5 Robustness to BEV Height

In Tab. 4, we ablate on the predefined height $z$ of BEV plane. When $z$ varies from $-1.0$ to $2.0$, the performance fluctuates little (smaller than $0.4$ mIoU). The predefined height $z$ affects the BEV-to-2D projection. But GKT only requires coarse projection and thus is quite insensitive to the value of $z$. The experiments prove the robustness of GKT.

| Kernel | $\sigma_1(m)$ | | | | |
|---|---|---|---|---|---|
| | 0 | 0.05 | 0.1 | 0.5 | 1.0 |
| $3 \times 3$ | 36.5 | $36.5^{\downarrow 0.0}$ | $36.3^{\downarrow 0.2}$ | $33.1^{\downarrow 3.4}$ | $27.3^{\downarrow 9.2}$ |
| $5 \times 5$ | 36.6 | $36.5^{\downarrow 0.1}$ | $36.4^{\downarrow 0.2}$ | $33.9^{\downarrow 2.7}$ | $28.8^{\downarrow 7.8}$ |
| $7 \times 3$ | 37.3 | $37.3^{\downarrow 0.0}$ | $37.1^{\downarrow 0.2}$ | $34.9^{\downarrow 2.4}$ | $30.5^{\downarrow 6.8}$ |

TABLE 2: **Robustness to the translation deviation of camera.** The metric is mIoU. $\sigma_1$ is the standard deviation of $\Delta_x, \Delta_y, \Delta_z$.

| Kernel | $\sigma_2(rad)$ | | | | |
|---|---|---|---|---|---|
| | 0 | 0.005 | 0.01 | 0.02 | 0.05 |
| $3 \times 3$ | 36.5 | $36.2^{\downarrow 0.3}$ | $35.5^{\downarrow 1.0}$ | $33.6^{\downarrow 2.9}$ | $25.6^{\downarrow 10.9}$ |
| $5 \times 5$ | 36.6 | $36.3^{\downarrow 0.3}$ | $35.7^{\downarrow 0.9}$ | $34.1^{\downarrow 2.5}$ | $27.4^{\downarrow 9.2}$ |
| $7 \times 3$ | 37.3 | $37.0^{\downarrow 0.3}$ | $36.5^{\downarrow 0.8}$ | $34.9^{\downarrow 2.4}$ | $28.4^{\downarrow 8.9}$ |

TABLE 3: **Robustness to the rotation deviation of camera.** The metric is mIoU. $\sigma_2$ is the standard deviation of $\theta_x, \theta_y, \theta_z$.

| Kernel | z=-1.0 | z=0.0 | z=1.0 | z=2.0 |
|---|---|---|---|---|
| $3 \times 3$ | 36.5 | 36.4 | 36.4 | 36.6 |
| $5 \times 5$ | 36.4 | 36.8 | 36.7 | 36.5 |

TABLE 4: **Robustness to the predefined height $z$ of BEV plane.** The metric is mIoU. The performance of GKT fluctuates little.

### 3.6 Convergence Speed

Fig. 3 compares the convergence speed between GKT and CVT [7]. We evaluate the models with different training schedules, *i.e.*, from 1-epoch to 8-epoch schedule. Based on the geometric priors, the proposed GKT achieves faster convergence speed than CVT and obtains much better results with only 1-epoch training.

### 3.7 Implementation of GKT

The proposed 2D-to-BEV transformation can be implemented in different manners.

1) Im2col: we first adopt `im2col` operation to unfold image features to the column formats. Each column corresponds to a kernel region. Then we select the corresponding column for each BEV query. Im2col is straightforward but results in large memory consumption.

2) Grid Sample: we adopt `grid_sample` operation to sample features of all pixels in the kernel regions, and concat them together.

3) LUT Indexing: to further accelerate the inference of transformation, we pre-compute the correspondence between indices of BEV grids and indices of image pixels and then building a look-up table.

Tab. 5 compares the inference speeds (FPS) of different implementation manners (kernel is $3 \times 3$). LUT Indexing achieves the best inference speed as expected.

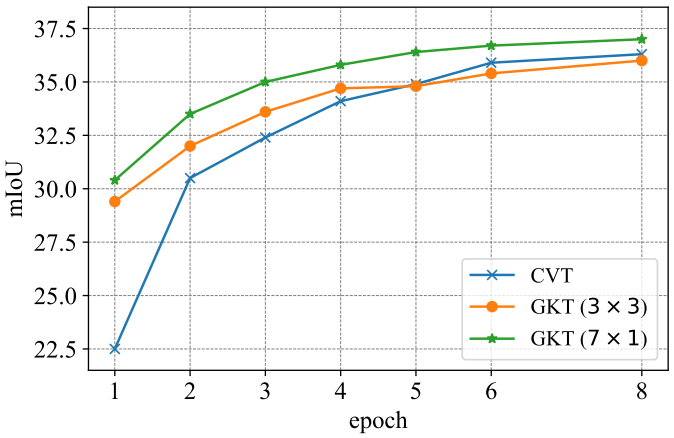



Fig. 3: **Comparison of the convergence speed.** We train GKT and CVT [7] with different training schedules. For only 1-epoch training schedule, the gap of mIoU is obvious. GKT converges much faster than CVT [7].

| Imp. | Im2col | Grid Sample | LUT Indexing |
|---|---|---|---|
| FPS | 38.8 | 42.3 | 43.4 |

TABLE 5: **Comparison of different implementation manners.** We adopt different implementations of the proposed GKT and evaluate the inference speed on one NVIDIA 2080Ti.

## 4 Related Work

### 4.1 BEV Representation Learning

Bird's Eye View is a promising representation in autonomous driving. Recent works study how to transform image representation to BEV representation. Lift-Splat [2], FIERY [1] and BEVDet [6] lift camera features to 3D by predicting a depth probability distribution over pixels and using known camera intrinsics and extrinsics. OFT [4] proposes orthographic feature transformation. BEVFormer [3] predicts height offsets of BEV anchor points and then use camera's parameters to fix the 2D-to-3D correspondence. PON [5] condenses the image features along the vertical dimension and then adopts dense transformer layer to expand features along the depth axis. The above methods all require precise calibrated camera's parameters for 2D-3D transformation. Differently, CVT [7] globally aggregates information from all image features through a series of cross attention layers to generate BEV representation, without explicitly adopting parameters. GKT is also robust to camera's parameters but achieves better efficiency by leveraging geometric priors.

### 4.2 Perception and Planning based on BEV

Previous works study the perception and planning based on Bird's Eye View representation. FIERY [1] predicts temporally consistent future instance segmentation and motion in bird's-eye view. [2], [5] conducts map segmentation based on BEV. [4], [6] adopts BEV representation for 3D object detection. BEVFormer [3] builds up detection and segmentation head on BEV feature maps for multi-task learning. GKT provides robust BEV representation. Various perception and planning tasks can be based on GKT.

## 5 Conclusion

We present GKT for 2D-to-BEV representation learning. GKT has high efficiency and robustness, both of which are crucial characteristics for a real-time running system, especially for autopilot. We validate GKT on map-view segmentation. In the near future, we will extend it to other BEV-based tasks, like detection and motion planning.

## References

[1] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "FIERY: future instance prediction in bird's-eye view from surround monocular cameras," in *ICCV*, 2021. 1, 4, 5

[2] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," in *ECCV*, 2020. 1, 4, 5

[3] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," *arXiv:2203.17270*, 2022. 1, 4, 5

[4] T. Roddick, A. Kendall, and R. Cipolla, "Orthographic feature transform for monocular 3d object detection," in *BMVC*, 2019. 1, 5

[5] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in *CVPR*, 2020. 1, 4, 5

[6] J. Huang, G. Huang, Z. Zhu, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," *arXiv:2112.11790*, 2021. 1, 5

[7] B. Zhou and P. Krähenbühl, "Cross-view transformers for real-time map-view semantic segmentation," *CVPR*, 2022. 1, 3, 4, 5

[8] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020. 3

[9] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019. 3

[10] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robotics Autom. Lett.*, 2020. 4

[11] A. Saha, O. M. Maldonado, C. Russell, and R. Bowden, "Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation," in *ICRA*, 2021. 4