A Novel Convolution and Attention Mechanism-based Model for 6D Object Pose Estimation

Alexander Du
Bellevue School District
s-dual@bsd405.org

Yingwu Zhu
Seattle University
zhuy@seattleu.edu

***Abstract*—Estimating 6D object poses from RGB images is challenging because the lack of depth information requires inferring a three-dimensional structure from 2D projections. Traditional methods often rely on deep learning with grid-based data structures but struggle to capture complex dependencies among extracted features. To overcome this, we introduce a graph-based representation derived directly from images, where each pixel's spatial-temporal features serve as nodes, and relationships between them are defined through node connectivity and spatial interactions. We also employ feature selection mechanisms that use spatial-attention and self-attention distillation, along with a Legendre convolution layer leveraging the orthogonality of Legendre polynomials for numerical stability. Experiments on the LINEMOD, Occluded LINEMOD, and YCB-VIDEO datasets demonstrate that our method outperforms nine existing approaches and achieves state-of-the-art benchmark in object pose estimation.**

I. Introduction

Accurate object identification and estimation of their three-dimensional position and orientation $(x, y, z, \theta, \psi, \phi)$ are critical tasks in computer vision and robotics, with applications ranging from autonomous driving to augmented reality and industrial automation. Traditional approaches to six degrees of freedom (6-DOF) pose estimation relied on matching feature points between model projections and camera images, followed by techniques like Perspective-n-Point for pose calculation. While effective for textured objects, these methods struggle with objects lacking distinctive surface features (Collet et al., 2011; Brachmann et al., 2014). The advent of depth-sensing technology alleviated some challenges by providing additional geometric data, but it was the integration of deep learning techniques, such as PoseCNN (Xiang et al., 2017), that marked a significant leap. Leveraging large datasets and advanced feature analysis, deep learning enabled pose estimation using only RGB images.

Despite these advancements, convolutional neural networks (CNNs), the backbone of many deep learning approaches, face limitations in capturing feature dependencies due to their grid-based data structures. To address these shortcomings, we propose **PoseLecTr**, a graph-based encoder-decoder architecture that represents image features as graph nodes and models their relationships using an adjacency matrix. By employing a graph neural network framework, *PoseLecTr* effectively maps feature dependencies and independencies, improving upon traditional CNN-based methods. Additionally, the integration of spatial attention mechanisms, self-attention distillation, and a Legendre convolution technique enhances computational efficiency and accuracy. These innovations position *PoseLecTr* as a robust solution for advancing 6-DOF pose estimation in challenging scenarios.

II. Related Works

The task of 6D object pose estimation, involving the determination of an object's orientation and position in three-dimensional space, is fundamental in robotics and computer vision. Traditional approaches relied on matching feature points between 2D images and 3D models, often requiring distinctive textures for accurate correspondences (Collet et al., 2011; Brachmann et al., 2014). With the rise of deep learning, feature-based methods have advanced pose estimation by leveraging keypoints and edges to align RGB images with 3D models, achieving robust results even in challenging scenarios (Zhu et al., 2021; Zhao et al., 2020; Peng et al., 2019). Template-based methods further extend this by precomputing 3D model templates and matching them to real-time images for pose inference (Ulrich et al., 2011; Sundermeyer et al., 2020).

Convolutional neural networks (CNNs) have significantly enhanced pose estimation by enabling direct predictions of object poses from RGB images. Methods such as PoseCNN (Xiang et al., 2017) predict 3D translations and rotations, often incorporating Perspective-n-Point (PnP) refinements to improve accuracy. However, grid-based CNNs face challenges in capturing higher-order dependencies, particularly in complex scenes.

To address these limitations, graph neural networks (GNNs) have emerged as a promising alternative for modeling non-Euclidean data. GNNs represent image features as nodes and their relationships as edges, enabling the aggregation of contextual information (Wu et al., 2020). Advances in GNNs include spectral methods for adaptive graph structures (Li et al., 2018a) and Cluster-GCN for efficient large-graph training (Chiang et al., 2019). Unsupervised approaches, such as graph autoencoders (Cao et al., 2016 [1]; Yu et al., 2018), further extend the representational capacity of GNNs, while attention-based methods like ASTGCN (Guo et al., 2019) integrate spatial-temporal attention for dynamic systems.

Although these approaches mark significant progress, many remain constrained by grid-based feature extraction. This study builds on existing graph-based methods by introducing a novel encoder-decoder framework with Legendre convolution

layers and attention mechanisms, improving the accuracy and robustness of 6D pose estimation.

III. METHODOLOGY

In this section, we introduce notation and formalize the problem of 6D object pose estimation from RGB images. We begin by modeling an image as a graph $G(V, \varepsilon, A)$, where $V = \{v_1, v_2, v_3, \ldots, v_N\}$ is the set of N nodes representing pixels, ε is a set of edges specifying connections among these nodes, and A denotes the adjacency matrix. The adjacency matrix is constructed via cosine similarity of feature vectors derived from the image, thereby capturing relational information among different regions.

Let I_t represent the input RGB image at time t. The goal is to estimate the 6D pose of the object in the image, comprising its 3D rotation and 3D translation. We represent the pose as a tensor P_t, encompassing pose information across multiple images.

The architecture we introduce, named **PoseLecTr**, is composed of several key components: a feature extraction layer, a Legendre-based convolution layer, a spatiotemporal encoding layer, and a final decoding layer. In the first stage, the feature extraction mechanism transforms the raw image data into complex high-dimensional features. The convolution process in the second layer is approximated through the use of Legendre polynomials, chosen for their properties of orthogonality, numerical stability, and recursive efficiency. The spatiotemporal encoder, which uses a self-attention mechanism, is responsible for capturing intricate spatiotemporal relationships, essential for interpreting both the object's movement and spatial structure. The decoder is tasked with producing the resulting 6D pose estimation. Fig. 1 illustrates the PoseLecTr framework, which includes the feature extraction module, several spatiotemporal encoding layers stacked in sequence, and the final output layer. A detailed explanation of each component follows.

A. Feature Extraction Layer

The feature extraction layer is modified to utilize the third convolutional block of the ResNet50 network, which helps reduce redundant features, compress the model size, and decrease the number of parameters. Specifically, the image is processed through the initial layers of ResNet50, up to the third convolutional block, to extract meaningful feature maps. By using this intermediate layer, as opposed to deeper layers, the model retains essential spatial and structural information while reducing unnecessary details, thereby decreasing computational complexity and avoiding overfitting. The extracted feature maps are then flattened and passed through a fully connected layer to obtain the final embedding, with an embedding dimension. The feature extraction layer converts the raw input image I_t into high-dimensional feature representations. This is achieved through a series of convolutional layers followed by fully connected layers. Specifically, the image is passed through convolutional layers to extract feature maps, which are then flattened and further processed through a fully connected layer to obtain the embedding, denoted as F_t with an embedding dimension d:

$$F_t = FC(Flatten(Conv(I_t))) \tag{1}$$

where $F_t \in R^{N \times d}$ represents the feature matrix, N is the number of pixels, and d is the embedding dimension.

B. Legendre Convolution Layer

The data embedding layer transforms the input into a high-dimensional representation. Initially, the raw input $\chi = (X_1, X_2, \ldots, X_T) \in R^{T \times N \times c}$ is processed through a fully connected layer to obtain $\chi = (X_1, X_2, \ldots, X_T) \in R^{T \times N \times d}$, where d is embedding dimension, T is the number of input, N is the number of pixels, c is the image channels. Subsequently, a spatial-temporal embedding mechanism is designed to integrate essential knowledge into the model. This mechanism includes the spatial graph Laplacian embedding for encoding the spatial relationships within the image and the temporal embedding for capturing the dynamic changes in object pose over time.

To characterize the interconnections between nodes in the graph, we utilize the regularized Laplacian matrix denoted as $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where A represents the adjacency matrix, I is the identity matrix, and D is the matrix of node degrees in diagonal form. As L is a symmetric positive semidefinite matrix, its eigenvectors are orthogonal in pairs, and the eigenvalues are distinct and non-negative. Therefore, L can be simplified to the form of:

$$L = U \, \Delta \, U^T \tag{2}$$

where U is a matrix composed of eigenvectors and Δ is a matrix composed of eigenvalues.

In the Graph Convolutional Network (GCN) model, Y_{output} is define as:

$$Y_{output} = \sigma \left(U \, \mathbf{g}_\theta(\Delta) \, U^T \chi \right) \tag{3}$$

This formula represents a layer in the standard GCN, where $\sigma()$ denotes the activation function, and θ is the parameter value for iterative updates. Here, $\mathbf{g}_\theta(\Delta)$ signifies the convolution kernel operating on a graph.

The number of convolution parameters is equivalent to the number of graph nodes, denoted as n. For large graphs, the computational efficiency of this method is significantly diminished. Additionally, the algorithm necessitates the computation of $\mathbf{g}_\theta(\Delta) U^T$ each time, resulting in a high time complexity of $O(n^3)$.

To address these challenges, we employ Legendre polynomials to approximate the convolution kernel. Legendre polynomials are defined as follows:

$$P_0(x) = 1$$

$$P_n(x) = \frac{1}{2^n \times n!} \cdot \frac{d^n}{dx^n} [(x^2 - 1)^n], \; n = 1, 2, \ldots \tag{4}$$

There are several advantages to employing Legendre polynomials for approximation:

1) Orthogonality: Legendre polynomials of different orders have a zero inner product, allowing each coefficient to be computed independently during polynomial fitting or series expansion. This property streamlines the overall computation process.
2) Numerical Stability: Legendre polynomials typically exhibit strong numerical stability in the fitting process,

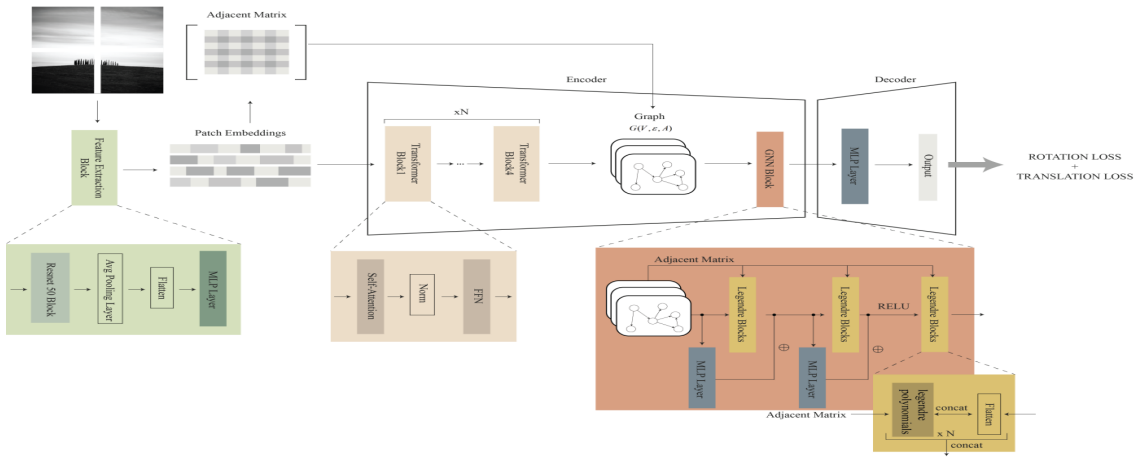

Fig. 1. Architecture of $PoseLecTr$

helping to mitigate undesirable artifacts such as Runge's phenomenon.

3) Recursion: Legendre polynomials satisfy a straightforward recursive relationship, enabling efficient generation of higher-order polynomials.

$$P_0(x) = 1 \tag{5}$$

$$P_1(x) = x \tag{6}$$

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x),\ n = 1, 2\ldots \tag{7}$$

After replacing the convolution kernel $\mathrm{g}_\theta(\Delta)$ with the Legendre polynomial, a new convolution kernel form is obtained:

$$\mathrm{g}_\theta(\Delta) = \sum_{k=0}^{n-1} \alpha_k P_k(\widetilde{\Delta}) \tag{8}$$

C. Encoder Layer

We have developed a encoder layer based on a self-attention mechanism to capture the intricate and dynamic dependencies required for accurate 6D object pose estimation from RGB images. The core of the encoder layer comprises two main components: the **spatial feature self-attention module** and the **self-attention distillation module**. These modules work together to effectively model the complex relationships present in the data. These modules collectively enable the encoder layer to effectively understand and model complex relationships in embedding data.

$$Q \in R^{L_Q \times d}, \quad K \in R^{L_K \times d}, \quad V \in R^{L_V \times d}$$

and d is input dimension. The self-attention mechanism has demonstrated effectiveness in various scenarios.

D. Spatial Features Self-attention

We design spatial feature self-attention to capture dynamic spatial features in RGB images for accurate 6D object pose estimation. Formally, for input $\chi = (X_1, X_2, \ldots, X_T) \in R^{T \times N \times d}$, the $query$, key, and $value$ matrices of self-attention operations can be represented as:

$$Q_t = X_t \ \mathrm{g}\ W_Q$$

$$K_t = X_t \ \mathrm{g}\ W_K$$

$$V_t = X_t \ \mathrm{g}\ W_V$$

where $W_Q, W_K, W_V \in R^{d \times d'}$ are learnable parameters and d' is the dimension of the $query$, key, and $value$ matrices. The self-attention mechanism SA can be expressed as:

$$SA\,(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d}})\ V \tag{9}$$

As can be seen from the formula, the spatial dependencies between nodes are different in different X_i.

The key to the attention mechanism is the mapping function, which encodes the relative importance of the input elements, mapping the values to probabilities. The commonly used Softmax function generates dense attention. However, in the task of 6D pose estimation from RGB images, the distribution of feature relevance can be highly imbalanced across different regions of the image. Therefore, we design a new spatial feature attention module SFA, utilizing the Sparsemax function [21], which is expressed as:

$$SFA\,(Q, K, V) = Sparsemax(\frac{QK^T}{\sqrt{d}})\ V \tag{10}$$

In contrast to the Softmax function, the Sparsemax function can generate sparse probability distributions. Sparsemax has the ability to selectively emphasize the most significant features while disregarding less relevant ones, thereby offering clearer decision boundaries. Moreover, as not all image regions exhibit high correlation, Sparsemax reduces the model's sensitivity to noise by assigning zero probability to these less relevant regions. Furthermore, when dealing with complex image data for 6D pose estimation, Sparsemax is more adept at handling the long-tail effects, ensuring the model focuses on the most critical spatial features.

E. Self-attention Distillation

While we employed the Sparsemax function to minimize the redundancy in feature mapping within the encoder, certain redundant features persist. To address this, we utilize a self-attention distillation module to identify the crucial features with dominance and create a centralized self-focused feature

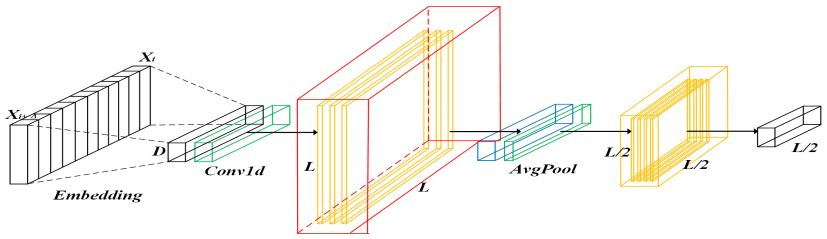

Fig. 2. Illustrates the structure within the encoder of the **PoseLecTr**. (1) This diagram represents one of several identical sections of the encode. (2) The primary section shown processes the entire input sequence. Subsequently, the next section handles only half of the input segments, and this pattern continues in the following sections. (3) The output of the encoder is formed by merging the feature maps from all sections.

map in the subsequent layer. This module effectively reduces the dimensionality of the input features, with the distillation process progressing from layer l to layer $l+1$ as follows:

$$X_{l+1} = AvgPool(ReLU(Conv1d([X_l]_{SFA}))) \tag{11}$$

where $[\ ]_{SFA}$ represents the spatial features self-attention block, $Conv1d()$ denotes a one-dimensional convolution operation. This includes multi-head spatial features self-attention and basic operations, where $Conv1d(\cdot)$ executes a one-dimensional convolution filter (kernel width = 3) in the time dimension using the $ReLU(\cdot)$ activation function. Following the stacking of one layer, we introduce an average-pooling layer with a stride of 3. Ultimately, the final feature representation of the encoder is obtained.

F. Abbreviations and Acronyms

Our methodology incorporates a traditional decoding architecture comprising two sequentially layered multi-headed attention mechanisms. Additionally, a bypass route, facilitated by 1 × 1 convolutional operations, is applied after each encoder layer to convert the outputs, labeled as χ_{output}, into a bypassed format $\chi_{sk} \in R^{T \times N \times d_{sk}}$, with d_{sk} representing the bypassed dimensionality.

IV. Experiments

A. Dataset Description

Three datasets—LINEMOD [46], Occluded LINEMOD [47], and YCB-Video [48]—were employed in the experiments, each posing distinct challenges reflective of real-world applications. LINEMOD provides a structured environment and annotated scenarios for controlled evaluations of object recognition and pose estimation. Occluded LINEMOD introduces significant occlusion and clutter, facilitating robust testing of algorithm resilience under partial visibility. By contrast, YCB-Video contains a wide range of object instances captured in dynamic settings, including robotic manipulation tasks, thereby assessing an algorithm's adaptability to both static and changing environments. Collectively, these datasets introduce dense and chaotic scenes, multiple objects, and varied environmental conditions—from static desktop configurations to dynamic robotic tasks—and encompass objects with untextured surfaces, smooth geometries, high degrees of symmetry, and different sizes and proportions.

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
002_master_chef_can	89.0	93.9	89.3	86.5	90.2	91.5	92.7	94.0	95.2	96.4
003_cracker_box	88.5	96.5	87.8	85.0	89.9	91.0	92.5	94.7	95.8	97.8
004_sugar_box	94.3	97.6	91.2	89.7	92.3	93.5	94.8	96.0	95.2	97.5
005_tomato_soup_can	89.1	95.0	88.9	86.8	90.7	91.9	93.2	94.5	95.7	98.5
006_mustard_bottle	92.0	95.8	90.5	88.6	91.8	93.0	94.2	95.5	96.7	97.3
007_tuna_fish_can	78.0	86.5	79.3	77.5	81.8	83.2	84.5	85.8	87.9	91.2
008_pudding_box	81.4	97.9	83.7	80.9	84.2	85.6	87.1	89.3	90.5	91.8
009_gelatin_box	90.4	97.8	92.1	89.8	93.0	94.2	95.6	96.7	97.9	98.7
010_potted_meat_can	81.7	77.8	80.3	78.2	82.1	83.5	84.9	86.3	87.5	90.2
011_banana	82.6	94.9	84.5	83.2	86.5	87.8	89.1	91.3	92.6	96.9
All Frames	86.66	93.37	86.76	84.62	88.25	89.52	90.86	92.41	93.61	95.63

TABLE I
POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD ON THE YCB-VIDEO DATASET.

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
002_master_chef_can	85.3	92.2	87.1	84.4	89.6	89.6	89.0	93.7	92.5	93.9
003_cracker_box	86.6	94.5	85.2	83.6	87.2	88.2	89.1	93.2	95.0	97.1
004_sugar_box	91.9	95.7	87.3	87.8	91.2	92.0	91.7	95.8	96.8	95.5
005_tomato_soup_can	87.3	94.5	87.8	83.6	89.9	90.1	90.0	93.2	93.6	96.7
006_mustard_bottle	91.6	91.9	89.2	86.1	90.3	92.3	92.6	95.0	93.5	96.3
007_tuna_fish_can	77.1	83.0	75.7	75.2	78.6	81.0	82.7	85.6	85.9	88.0
008_pudding_box	77.5	96.6	83.2	79.3	81.7	85.1	85.2	87.3	89.0	89.8
009_gelatin_box	90.0	94.9	88.6	89.7	91.0	93.0	93.2	93.7	95.5	95.8
010_potted_meat_can	77.9	74.4	78.9	76.9	79.9	80.2	83.5	82.6	85.1	88.2
011_banana	80.1	93.5	81.1	79.6	84.5	84.4	88.2	88.1	89.4	95.2
All Frames	84.53	91.12	84.41	82.62	86.39	87.59	88.52	90.82	91.63	93.60

TABLE II
POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD-S ON THE YCB-VIDEO DATASET.

B. Experimental Details

Baselines: We evaluated **PoseLecTr** against nine baseline models spanning three broad categories:

CNN-based Models PoseCNN[48] directly predicts the 3D translation and rotation from RGB images, establishing an early benchmark in integrating RGB and depth data for high-precision 6D pose estimation. DenseFusion[49] fuses dense pixel-wise representations from both RGB and depth maps, achieving accurate 6D pose predictions by leveraging the complementary information of the two modalities. PVNet[50] regresses 2D keypoints from images and applies a PnP algorithm to calculate the final pose, thus improving accuracy through precise keypoint localization. PoseRBPF[51] employs recurrent neural networks coupled with particle filters to handle temporal dynamics, reinforcing robustness against occlusion and temporal inconsistencies in pose estimation. DeepIM[52] refines an initial pose estimate in an iterative process by aligning rendered and observed images, creating a feedback loop that improves pose estimation accuracy over successive iterations.

GNN-based Models GC-Pose[53] employs graph convolutional networks to model spatial relationships among object parts, extracting features from RGB images to represent the object's geometric structure. G2LNet[54] similarly uses a graph neural network (GNN) to learn spatial relationships among keypoints, using these learned relationships to predict the 6D pose. While both methods leverage GNNs to capture spatial relationships, GC-Pose focuses on constructing a graph of object parts, whereas G2LNet relies on detected keypoints

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
ape	90.4	96.6	92.4	89.1	91.0	93.4	95.2	96.0	97.1	98.9
benchvise	89.8	98.0	89.2	86.7	90.8	94.1	93.4	96.6	97.5	99.6
cam	96.6	99.8	92.6	91.7	94.3	94.4	97.3	97.3	95.9	98.4
can	89.8	96.1	90.8	89.3	93.4	92.5	94.4	96.4	97.5	100
cat	91.0	97.8	89.9	85.7	91.1	93.4	94.6	96.7	97.4	99.4
driller	96.9	100.3	92.7	92.2	95.5	96.0	96.9	97.8	96.6	99.1
duck	91.5	96.0	90.5	88.3	92.4	93.3	94.3	95.4	97.6	100
eggbox	94.5	96.9	92.0	90.2	93.1	95.5	96.1	98.1	98.6	99.5
glue	79.2	87.8	80.8	79.4	83.5	85.8	86.6	88.8	89.1	93.1
holepuncher	82.2	100.7	85.6	83.5	85.2	87.4	89.1	91.0	93.4	94.7
iron	91.7	100.3	95.0	91.1	94.2	95.9	97.8	98.4	99.3	99.3
lamp	82.8	79.4	83.1	80.5	83.6	85.9	86.6	88.6	88.3	91.1
phone	83.4	96.7	86.0	83.8	87.6	90.1	90.9	93.1	94.1	99.6
All Frames	88.9	95.4	88.2	86.3	90.8	91.1	92.2	93.9	95.5	97.7

TABLE III

POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD ON THE LINEMOD DATASET.

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
ape	90.3	95.7	91.3	88.4	90.8	92.8	94.5	95.8	96.8	98.0
benchvise	89.6	97.7	88.6	86.0	90.4	93.0	92.9	96.4	96.6	99.4
cam	96.2	98.7	91.7	91.6	93.2	94.1	96.6	96.3	95.8	98.3
can	89.7	95.4	90.1	88.2	92.3	92.4	94.0	95.6	97.2	99.9
cat	89.9	97.1	89.7	85.3	91.1	92.7	93.5	96.6	96.3	98.8
driller	96.2	99.2	91.6	91.6	94.3	95.4	95.9	97.0	96.4	98.8
duck	90.9	95.6	89.9	88.1	92.1	92.2	94.1	94.9	96.8	99.3
eggbox	93.9	96.8	91.4	90.0	92.1	94.3	95.5	97.3	97.5	98.9
glue	78.5	87.6	80.2	78.7	82.5	85.1	85.5	87.7	88.7	92.7
holepuncher	82.1	99.6	85.6	82.7	85.1	86.9	88.4	90.1	92.4	93.7
iron	90.9	99.3	94.1	91.0	93.4	95.5	97.1	98.3	98.8	99.2
lamp	82.2	78.2	82.0	79.6	82.6	85.3	85.6	87.8	88.2	91.1
phone	83.3	96.5	85.2	83.5	87.1	89.2	90.5	92.2	93.9	98.6
All Frames	88.5	94.6	87.7	86.0	90.0	90.9	91.4	93.5	94.9	97.1

TABLE IV

POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD-S ON THE LINEMOD DATASET.

to guide the pose estimation process.

Self-attention-based Models TransPose[55] adopts a transformer based architecture to model long-range dependencies in image data, leveraging self-attention mechanisms to focus on image regions most relevant to 6D pose prediction. PoseFormer[56] combines convolutional neural networks (CNNs) for local feature extraction with transformers to capture global context, thereby producing high-precision 6D pose estimates. While both methods integrate transformers to encode dependencies and contextual information, TransPose emphasizes self-attention to isolate key regions for pose estimation, whereas PoseFormer leverages CNNs for local features and transformers for a broader, global perspective.

C. Hyper-parameter Settings

Our proposed methods are trained using the Adam optimizer, with an initial learning rate of 1×10^{-4}, which is reduced by half after each epoch. The training process spans 8 epochs, incorporating early stopping to prevent overfitting. All baseline methods are configured according to their recommended settings, using a batch size of 32.

D. Evaluation Metrics

In our research, we applied two distinct indicators[48] to measure performance:

$$ADD = \frac{1}{m} \sum_{x \in M} \|(Rx + t) - (\widehat{R}x + \widehat{t})\| \tag{12}$$

$$ADDS = \frac{1}{m} \sum_{x_1 \in M} \min_{x_2 \in M} \|(Rx_1 + t) - (\widehat{R}x_2 + \widehat{t})\| \tag{13}$$

The ADD metric quantifies the accuracy of the predicted attitude by measuring the mean Euclidean distance between the corresponding three-dimensional points of the model. If the average distance between these points is less than 10%, the posture is considered correct. Similarly, for symmetric objects, where multiple gestures may correspond to the same physical configuration, ADD-S metrics are used. Instead of calculating point-to-point distance, ADD-S measures the average minimum distance between each transformation point in the predicted attitude and the nearest point on the model transformed by the ground true state. If this average minimum distance is less than 10%, the posture is considered correct.

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
ape	86.8	91.6	89.0	84.1	88.1	90.6	91.2	93.1	93.4	93.7
can	85.5	93.2	86.2	83.3	89.1	89.8	91.8	92.3	94.9	95.5
cat	91.6	96.1	90.5	89.3	90.4	91.9	91.7	94.4	94.7	95.5
driller	87.3	94.6	86.5	86.2	89.5	90.7	91.8	94.2	93.8	94.3
duck	86.2	93.9	87.3	83.6	89.8	90.3	90.4	92.8	93.1	93.8
eggbox	93.2	96.9	89.0	88.5	90.8	90.9	92.8	93.6	94.6	95.2
glue	87.6	92.6	87.9	83.7	88.7	90.6	91.5	92.5	92.6	92.8
holepuncher	91.0	93.6	89.2	87.6	90.4	91.5	92.6	93.5	93.8	95.0
All Frames	88.6	94.0	88.2	85.7	89.6	90.7	91.7	93.3	93.8	94.4

TABLE V

POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD ON THE OCCLUDED LINEMOD DATASET.

Method	PoseCNN	DenseFusion	PVNet	PoseRBPF	DeepIM	GC-Pose	G2LNet	TransPose	PoseFormer	Ours
ape	85.5	90.0	87.9	82.9	86.4	89.5	89.8	91.8	91.9	92.3
can	83.9	91.9	85.0	82.1	87.8	88.5	90.5	90.6	93.7	94.3
cat	89.9	94.7	88.9	88.0	89.4	90.2	90.1	92.9	93.1	93.7
driller	85.6	93.3	84.9	84.6	87.9	89.2	90.2	92.6	92.6	92.9
duck	84.4	92.3	85.6	82.1	88.6	88.9	89.1	91.5	91.6	92.5
eggbox	92.2	95.4	87.9	87.3	89.1	89.2	91.8	92.4	93.6	93.6
glue	86.4	91.0	86.5	82.6	87.1	89.4	89.9	91.5	90.9	91.4
holepuncher	89.8	92.6	87.9	86.4	89.2	90.3	90.8	92.3	92.6	93.8
All Frames	87.2	92.6	86.8	84.5	88.1	89.4	90.2	91.9	92.5	93.0

TABLE VI

POSE TRACKING PERFORMANCE OF RGB-D METHODS EVALUATED USING ADD-S ON THE OCCLUDED LINEMOD DATASET.

E. Experiment Platform

All experiments were conducted on a workstation equipped with an NVIDIA GeForce RTX 4090 and 64 GB of RAM, running Ubuntu 18.04. The $PoseLecTr$ was implemented in PyTorch 1.16.10 using Python 3.9.0.

F. Performance Comparison

For the object $002\_master\_chef\_can$, *PoseLecTr* outperforms all methods in these two tables, achieving 96.4% and 93.9% accuracy in Tables I and II, respectively. On $003\_cracker\_box$, *PoseLecTr* consistently shows superior performance at 97.8% and 95%, while DenseFusion follows with 96.5% and 94.5%. More challenging objects, such as $007\_tuna\_fish\_can$, also highlight *PoseLecTr*'s advantages, with accuracies of 91.2% in Table I and 88% in Table II. Considering the overall performance across all frames, *PoseLecTr* achieves the highest accuracy in both tables, scoring 95.63% (Table I) and 93.36% (Table II), indicating consistent superiority over diverse object categories and frameworks. TransPose and PoseFormer follow closely; in Tables I and II, TransPose registers 92.41% and 90.82%, whereas PoseFormer achieves 93.61% and 91.63%.

In Tables III and IV, PoseLecTr demonstrates the best pose tracking performance on the LINEMOD dataset, achieving overall accuracies of 97.7% and 97.1% under the ADD and ADD-S metrics, respectively—higher than other methods. On the more challenging Occluded LINEMOD dataset (Tables V and VI), *PoseLecTr* maintains its robustness by posting accuracies of 94.4% and 93.0%, consistently leading the benchmarks.

G. Ablation Study

To further investigate the efficacy of **PoseLecTr**'s components, we conducted a comparative analysis using several modifications of *PoseLecTr*, resulting in three different variants:

PoseLecTr+: In this variant, *Legendre convolution* is replaced by *Chebyshev convolution* to assess the impact of different graph convolution modes.

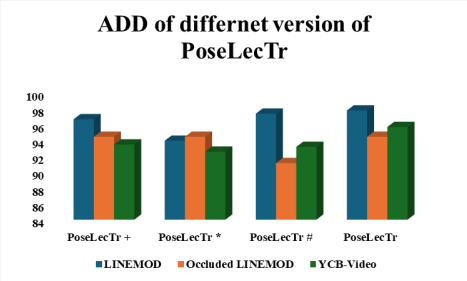

Fig. 3. Ablation Study: Impact of PoseLecTr Variants on ADD in 3 datasets

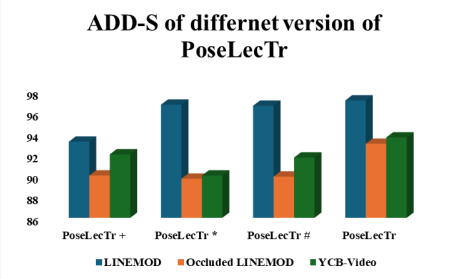

Fig. 4. Ablation Study: Impact of PoseLecTr Variants on ADD-S in 3 datasets

PoseLecTr*: This version removes the *spatial features self-attention* mechanism, causing every node in the graph to be influenced uniformly by all other nodes.
PoseLecTr#: This adjustment excludes the module responsible for *distilling self-attention*, hindering the processing of extended data, particularly when the graph size is large.

Fig. 3 and Fig. 4 illustrate the comparative performance of these variants on the YCB-Video, LINEMOD, and Occluded LINEMOD datasets. From these results, we can derive the following insights:

1. **Effect of Graph Convolution Type(PoseLecTr+)**: Replacing Legendre convolution with Chebyshev convolution in PoseLecTr+ results in slightly lower performance across all datasets. On the LINEMOD dataset, accuracy decreases from 99.7% (PoseLecTr) to 96.4%, with similar declines on Occluded LINEMOD (94.34% vs. 95.54%) and YCB-Video (91.56% vs. 93.36%). These findings suggest that Legendre convolution more effectively captures both local and global contexts compared to Chebyshev convolution.
2. **Effects of Spatial Feature Self-attention(PoseLecTr*)**: When the spatial self-attention mechanism is removed from PoseLecTr*, the accuracy on Occluded LINEMOD (92.54%) and YCB-Video (89.96%) decreases more substantially. This indicates that spatial self-attention is particularly important in handling occlusion and complex scenes. Meanwhile, the performance on LINEMOD (96.9%) remains relatively stable, suggesting that removing spatial self-attention chiefly affects more challenging datasets where spatial dependencies play a more critical role.
3. **Effects of Self-attentional Distillation(PoseLecTr#)**: Excluding self-attentional distillation from PoseLecTr# modules leads to a decline in performance, particularly on LINEMOD (92.64%) and YCB-Video (89.56%). These results indicate that self-focused distillation is essential for handling larger graph sizes and extended data. Without this component, the model's capacity to manage extensive spatial relationships is diminished, thereby reducing its ability to generalize to more complex environments.

V. Conclusion

In this paper, we presented **PoseLecTr**, a novel graph-based framework for 6D object pose estimation from RGB images. By incorporating Legendre convolution and attention mechanisms, PoseLecTr advances both accuracy and efficiency in pose estimation tasks. The introduction of Legendre convolution, with its orthogonality and numerical stability properties, streamlines the convolution process and enables more efficient feature extraction compared to traditional convolution methods. Additionally, the spatial feature self-attention and self-attention distillation modules enhance the model's ability to focus on critical spatial-temporal features, thus improving the robustness and accuracy of pose estimation. Our evaluation on benchmark datasets such as LINEMOD, Occluded LINEMOD and YCB-Video demonstrates that PoseLecTr consistently outperforms existing state-of-the-art models, particularly in complex scenes with textureless, symmetric, or highly occluded objects.

REFERENCES

[1] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016, vol. 30.

[1] Amini, A.; Selvam Periyasamy, A.; and Behnke, S. YOLOPose: Transformer-based multi-object 6D pose estimation using keypoint regression. In *International Conference on Intelligent Autonomous Systems*, 392–406, Springer, 2022.

[2] Atwood, J.; and Towsley, D. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016

[3] Brachmann, E.; Krull, A.; Michel, F.; Gumhold, S.; Shotton, J.; and Rother, C. Learning 6d object pose estimation using 3d object coordinates. In *Computer Vision–ECCV 2014: 13th European Conference*, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13, 536–551. Springer, 2014.

[4] Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013.

[5] Cao, S.; Lu, W.; and Xu, Q. Deep neural networks for learning graph representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[6] Chen, W.; Jia, X.; Chang, H. J.; Duan, J.; and Leonardis, A. G2l-net: Global to local network for real-time 6d pose estimation with embedding vector features. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4233–4242, 2020.

[7] Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 257–266, 2019.

[8] Collet, A.; Martinez, M.; and Srinivasa, S. S. The moped framework: Object recognition and pose estimation for manipulation. *The international journal of robotics research*, 30(10): 1284–1306, 2011.

[9] Dai, H.; Kozareva, Z.; Dai, B.; Smola, A.; and Song, L. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, 1106–1114. PMLR, 2018.

[10] De Cao, N.; and Kipf, T. MolGAN: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973, 2018.

[11] Deng, X.; Mousavian, A.; Xiang, Y.; Xia, F.; Bretl, T.; and Fox, D. PoseRBPF: A Rao–Blackwellized particle filter for 6-D object pose tracking. *IEEE Transactions on Robotics*, 37(5): 1328–1342, 2021.

[12] Gallicchio, C.; and Micheli, A. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, 1–8. IEEE, 2010.

[13] Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 922–929, 2019.

[14] Henaff, M.; Bruna, J.; and LeCun, Y. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163, 2015.

[15] Hinterstoisser, S.; Lepetit, V.; Ilic, S.; Holzer, S.; Bradski, G.; Konolige, K.; and Navab, N. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision*, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11, 548–562. Springer, 2013.

[16] Josifovski, J.; Kerzel, M.; Pregizer, C.; Posniak, L.; and Wermter, S. Object detection and pose estimation based on convolutional neural networks trained with synthetic data. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 6269–6276. IEEE, 2018.

[17] Li, R.; Wang, S.; Zhu, F.; and Huang, J. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018a.

[18] Li, Y.; Tarlow, D.; Brockschmidt, M.; and Zemel, R. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493, 2015.

[19] Li, Y.; Vinyals, O.; Dyer, C.; Pascanu, R.; and Battaglia, P. Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324, 2018b.

[20] Li, Z.; Li, X.; Chen, S.; Du, J.; and Li, Y. SaMfENet: Self-Attention Based Multi-Scale Feature Fusion Coding and Edge Information Constraint Network for 6D Pose Estimation. *Mathematics*, 10(19): 3671, 2022.

[21] Martins, A.; and Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, 1614–1623. PMLR, 2016

[22] Marullo, G.; Tanzi, L.; Piazzolla, P.; and Vezzetti, E. 6D object position estimation from 2D images: A literature review. *Multimedia Tools and Applications*, 82(16): 24605–24643, 2023.

[23] Massa, F.; Russell, B. C.; and Aubry, M. Deep exemplar 2d-3d detection by adapting from real to rendered views. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6024–6033, 2016.

[24] Micheli, A. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3): 498–511, 2009.

[25] Oberweger, M.; Rad, M.; and Lepetit, V. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*, 119–134, 2018.

[26] Payet, N.; and Todorovic, S. From contours to 3d object detection and pose estimation. In *2011 International Conference on Computer Vision*, 983–990. IEEE, 2011.

[27] Peng, S.; Liu, Y.; Huang, Q.; Zhou, X.; and Bao, H. Pvnet: Pixel-wise voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4561–4570, 2019.

[28] Periyasamy, A. S.; Capellen, C.; Schwarz, M.; and Behnke, S. ConvPoseCNN2: prediction and refinement of dense 6D object pose. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics*, 353–371. Springer, 2020.

[29] Rad, M.; and Lepetit, V. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *Proceedings of the IEEE international conference on computer vision*, 3828–3836, 2017.

[30] Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks*, 20(1): 61–80, 2008.

[31] Sundermeyer, M.; Marton, Z.-C.; Durner, M.; and Triebel, R. Augmented autoencoders: Implicit 3d orientation learning for 6d object detection. *International Journal of Computer Vision*, 128(3): 714–729, 2020.

[32] Ulrich, M.; Wiedemann, C.; and Steger, C. Combining scale-space and similarity-based aspect graphs for fast 3D object recognition. *IEEE transactions on pattern analysis and machine intelligence*, 34(10): 1902–1914, 2011.

[33] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[34] Wang, C.; Xu, D.; Zhu, Y.; Mart´ın-Mart´ın, R.; Lu, C.; Fei-Fei, L.; and Savarese, S. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 3343–3352, 2019.

[35] Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24, 2020.

[36] Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121, 2019.

[37] Xiang, Y.; Schmidt, T.; Narayanan, V.; and Fox, D. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv:1711.00199, 2017.

[38] Xu, B.; and Chen, Z. Multi-level fusion based 3d object detection from monocular images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2345–2353, 2018.

[39] Xu, Y.; Lin, K.-Y.; Zhang, G.; Wang, X.; and Li, H. Rnnpose: Recurrent 6-dof object pose refinement with robust correspondence field estimation and pose optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 14880–14890, 2022.

[40] Yan, S.; Xiong, Y.; and Lin, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[41] You, J.-K.; Hsu, C.-C. J.; Wang, W.-Y.; and Huang, S.-K. Object pose estimation incorporating projection loss and discriminative refinement. *IEEE Access*, 9: 18597–18606, 2021.

[42] Yu, W.; Zheng, C.; Cheng, W.; Aggarwal, C. C.; Song, D.; Zong, B.; Chen, H.; and Wang, W. Learning deep network representations with adversarially regularized autoencoders. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2663–2671, 2018.

[43] Zhao, H.; Wei, S.; Shi, D.; Tan, W.; Li, Z.; Ren, Y.; Wei, X.; Yang, Y.; and Pu, S. Learning symmetry-aware geometry correspondences for 6d object pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14045–14054, 2023.

[44] Zhao, W.; Zhang, S.; Guan, Z.; Zhao, W.; Peng, J.; and Fan, J. Learning deep network for detecting 3d object keypoints and 6d poses. In *Proceedings of the IEEE CVF Conference on computer vision and pattern recognition*, 14134–14142, 2020.

[45] Zhu, Y.; Wan, L.; Xu, W.; and Wang, S. ASPP-DF-PVNet: atrous Spatial Pyramid Pooling and Distance-Filtered PVNet for occlusion resistant 6D object pose estimation. *Signal Processing: Image Communication*, 95: 116268, 2021.

[46] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *International Conference on Computer Vision (ICCV)*, pages 858–865, 2011.

[47] Eric Brachmann, Alexander Krull, Frank Michel, S. Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D object pose estimation using 3d object coordinates. In *13th European Conference on Computer Vision (ECCV)*, pages 536–551, 2014. 6, 7

[48] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018. 2, 6, 7

[49] Wang, Yisheng, et al. DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[50] Peng, Song, et al. "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[51] Drost, Bertram, et al. "PoseRBPF: A Rao-Blackwellized Particle Filter for 6D Object Pose Estimation." *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[52] Li, Yisheng, et al. "DeepIM: Deep Iterative Matching for 6D Pose Estimation." *European Conference on Computer Vision (ECCV)*, 2018.

[53] Bogdan, Panait, et al. "GC-Pose: Geometric and Correspondence based Graph Neural Network for 6D Pose Estimation." *Proceedings of IEEE Conference on Robotics and Automation (ICRA)*, 2020.

[54] Wen, Bowen, et al. "G2LNet: Global to Local Networks for Real-time 6D Pose Estimation with Embeddings." *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[55] Yang, Zhenyu, et al. "TransPose: Transformer for 6D Object Pose Estimation." *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[56] Zhou, Yufei, et al. "PoseFormer: When Transformers Meet 6D Object Pose Estimation." *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[57] P. Yin and J. Ye and G. Lin and Q. Wu "Graph neural network for 6D object pose estimation" *Knowledge-Based Systems*, Vol 218, 2021.

[58] C. Zheng and S. Zhu and M. Mendieta and T. Yang and C. Chen and Z. Ding "3D Human Pose Estimation with Spatial and Temporal Transformers" *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.