ADVANCED INTELLIGENT SYSTEMS
Open Access
www.advintellsyst.com

A Transformer-Based Network for Full Object Pose Estimation with Depth Refinement

Mahmoud Abdulsalam, Kenan Ahiska, and Nabil Aouf*

In response to increasing demand for robotics manipulation, accurate vision-based full pose estimation is essential. While convolutional neural networks-based approaches have been introduced, the quest for higher performance continues, especially for precise robotics manipulation, including in the Agri-robotics domain. This article proposes an improved transformer-based pipeline for full pose estimation, incorporating a Depth Refinement Module. Operating solely on monocular images, the architecture features an innovative Lighter Depth Estimation Network using a Feature Pyramid with an up-sampling method for depth prediction. A Transformer-based Detection Network with additional prediction heads is employed to directly regress object centers and predict the full poses of the target objects. A novel Depth Refinement Module is then utilized alongside the predicted centers, full poses, and depth patches to refine the accuracy of the estimated poses. The performance of this pipeline is extensively compared with other state-of-the-art methods, and the results are analyzed for fruit picking applications. The results demonstrate that the pipeline improves the accuracy of pose estimation to up to 90.79% compared to other methods available in the literature.

1. Introduction

Full object pose estimation is a crucial topic to address in the robotics domain. The ability to perceive the position of an object from a single modality such as Red,Green,Blue (RGB) image can find a broad application area including robotics for grasping tasks,[1] autonomous driving,[2] space applications[3] and robotics for virtual and augmented reality applications.[4] Although full (6D) pose estimation from a single image is intriguing, challenges like object appearance and texture, lighting conditions and object occlusion affect the estimation performance dramatically.[5]

Conventionally, a 6D object pose estimation problem is formulated as a feature mapping problem where feature points of 3D objects are matched on 2D images.[6–8] However, these methods are unable to detect features on smooth objects with minimum or no texture. Introducing an additional modality such as depth data has been used to solve the problem of features on texture-less objects,[9–11] in other words, a performance improvement is achieved once more data is available as in the form of RGB-D images.

With the emergence of convolutional neural networks (CNNs), some research leveraged this powerful tool as part of their pipeline to estimate 6D poses.[5,12] Compared to CNNs, transformer models are rapidly emerging as substitutes with higher efficiency and accuracy.[13–16] Thus, few pipelines adopting transformer-based models for 6D pose estimation in quest for better accuracy[17–19] exist.

In this article, we propose a novel transformer-based network for 6D object pose estimation (TransPose) where we aim to achieve a significant accuracy improvement compared to the existing methods. We introduce an improved transformer-based 6D pose estimation network with a novel depth refinement module. The pose estimation domain has several research gaps, including performance, multiple modalities, occlusion handling, and issues related to datasets and labeling. Among these, this article focuses on improving accuracy while using a single modality. Our model uses only a single RGB image, with no supplementary data such as a thermal image or depth information, and it produces a high- accuracy 3D translation and rotation estimation. For the initial pose estimations, we adapted the Detection Transformer (DETR) framework,[13] to directly regress the center of the target object from the RGB input image. Then an image patch of the target object is extracted, and the translation and rotation can directly be regressed by formulating additional prediction heads on DETR.[17] Indeed, feed-forward heads are added to regress the two components of the 6D pose (3D translation and 3D rotation). A novel depth refinement module is also introduced in our estimation pipeline to increase the accuracy of the pose estimation.

The architecture of the proposed method inherits two interdependent tasks to obtain the final 6D pose of the target object.

M. Abdulsalam, N. Aouf
School of Science and Technology
City University of London
London ECV1 0HB, UK
E-mail: mahmoud.abdulsalam@city.ac.uk

K. Ahiska
Defence Systems Technologies Department
Aselsan Inc.
Ankara, Turkey

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202400110.

DOI: 10.1002/aisy.202400110

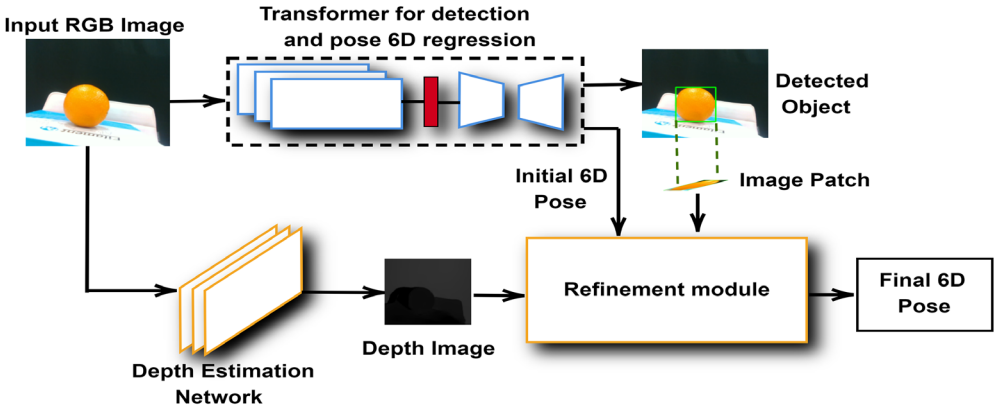

Figure 1. Overall solution architecture performing object detection, depth prediction, and 6D pose prediction.

As seen in **Figure 1**, an RGB image is used as the input to the pipeline. The image is passed to the transformer network which has a ResNet-101[20] backbone for feature extraction. These features are then passed to the transformer model consisting of a standard encoder and decoder setup.[21]

The model is used to obtain an image patch by detecting the object and assigning a Region Of Interest (ROI) to the detected object. The second segment of the architecture is the depth estimation and refinement module. The depth estimation network encompasses a feature pyramid network (FPN)[22] that takes in an RGB image as input and outputs an estimated depth image. The image patch obtained from the transformer model is used to isolate the target on the depth image and hence obtain the depth of the target from the camera. The depth is then used to compute other components of the translation and subsequently used to refine the estimated 6D pose of the target. We evaluated the performance of our model on the YCB-Video dataset[5] as a benchmark and compared it with the performance of other state-of-the-art approaches. The contributions introduced with our model can be listed as follows: 1) A novel pipeline for 6D object pose prediction is designed to favorably compare with other state-of-the-art methods, offering improved accuracy and robustness; 2) As part of the pipeline, we introduce an innovative lighter depth estimation network. This network utilizes a superior up-sampling method for depth prediction, enabling accurate depth estimation from a single monocular image. This approach completely eliminates the need for additional modalities/sensors such as depth images/cameras in pose estimation; and 3) We conducted extensive analyses using a custom-generated fruit dataset specifically designed to evaluate 6D pose estimation performance in fruit-picking applications.

The article continues with a literature review in Section 2. After introducing the solution for 6D pose estimation in Section 3, the details of the experimental work and the results are explained in Section 4 and finally, the article concludes with Section 5.

2. Related Work

Many methods have been proposed to address the challenge of 6D object pose estimation, utilizing various techniques and approaches. Non-learning-based methods traditionally rely heavily on object textures for accurate pose estimation. Examples include Scale-Invariant Feature Transform (SIFT)[23] and Speeded Up Robust Features (SURF),[24] which require rich texture information to perform effectively. However, these methods can struggle with texture-less objects. Miyake et al.[25] improved upon this limitation by incorporating color information to enhance pose estimation accuracy. Geometric information has also been leveraged to improve estimation precision.[26]

Pose estimation approaches that utilize local descriptors compute global descriptors offline, and then match local descriptors online for pose estimation. Techniques such as Iterative Closest Point (ICP), Oriented FAST and Rotated BRIEF (ORB),[27] and Binary Robust Independent Elementary Features (BRIEF)[28] fall under this category.[29–31] However, these methods are computationally expensive and may struggle with reflective surfaces.

Pose estimation methods can be categorized into template-based and feature-based approaches.[5] Template-based methods excel at detecting low-textured objects by matching input image locations with pre-constructed templates of the object.[10,32,33] However, they often fail to accurately estimate occluded objects due to low similarity scores in such cases. Feature-based methods, on the contrary, utilize 2D-3D correspondences to estimate poses,[6,8,34] offering better handling of occlusions but requiring rich feature information which can be lacking in texture-less objects. Recent advancements include learning-based feature descriptors[35,36] and direct regression from 2D images to 3D correspondences.[11,17,37,38]

Some recent models incorporate classical algorithms such as the Perspective-n-Point (PnP) algorithm to enhance pose estimation accuracy.[39–41] However, these models are often computationally intensive and may not be suitable for real-time applications. Models such as PoseCNN[5] and T6D-direct[17] achieve pose regression but typically require extensive training datasets and lack dedicated refinement modules.

CNN architectures like PoseNet[42] have shown promise in regressing 6D poses from RGB images. However, the absence of depth information can limit accuracy, motivating methods like depth prediction from 2D images to derive 3D positions.[5] Challenges in handling rotation components and post-refinement for accurate estimation have been addressed through

separate treatments.[43–45] Key-point detection methods[39,46] have also been proposed, using segmentation techniques or frameworks like YOLO,[47] but they may struggle with occlusions and truncations. Other pose estimation methods that utilize depth modalities typically involve converting depth images into point clouds for segmentation and subsequent pose regression.[48–50] This approach is computationally demanding and necessitates large datasets.

The transformer excels over CNNs by effectively capturing spatial relationships and dependencies among object keypoints without being limited to predefined local receptive fields. The ability to incorporate global context is pivotal for accurately estimating intricate object poses from images. Moreover, a pipeline may benefit from integrating a depth refinement module to enhance pose estimation accuracy, utilizing depth information derived directly from RGB images, thereby avoiding the need for conversion to point clouds as seen in related literature.

3. TransPose

The pipeline for TransPose 6D object pose estimation solution can be divided into three main parts: 1) Detection and Regression Transformer; 2) Depth Estimation Network (DEN); and 3) Refinement Module for Final 6D Pose Estimation.

3.1. Detection and Regression Transformer

This transformer network is mainly adopted for object detection, image patch designation and initial 6D pose regression. The transformer architecture is inspired by Detection Transformer DETR[13] and T6D-Direct.[17] As presented in **Figure 2**, an RGB image is used as the input of the model. A ResNet-101 is adopted as the CNN backbone to extract and create a feature vector which is used as an input to the transformer encoder-decoder. Set of predictions of size N_c is produced by the transformer encoder-decoder. Prediction heads are added in the form of Feed Forward Networks (FFNs) to regress the object pose and patch. The losses adopted to train this transformer network are categorized as follows:

3.1.1. Set Prediction Loss

The patch prediction in the form of region of interest (ROI) is obtained by assigning a bounding box around the object of interest. From the input image passed through the decoder, the model produces a set of tuples with fixed cardinality, where the size of the set is N_c which corresponds to the maximum number of the expected targets within the image. The content of each tuple is an image patch (left bottom pixel coordinates, height and width), class label probabilities and 6D pose (translation and rotation) of the predicted object. A bipartite matching is adopted to associate the ground truth and the predicted sets to obtain matching pairs. The model is then trained to minimize a loss between the pairs.

Consider ground truth objects $x_1, x_2, x_3, \ldots x_n$, let's assume N_c is more than the number of objects in the image, bipartite matching is performed to match the ground truth x which is a set of size N_c padded with no-object ($\varnothing$) with the predicted set $\hat{x}$of the same size. Essentially, performing a permutation between the sets while minimizing the loss in (1).

$$\hat{\rho} = \underset{\rho \in \Theta_{N_c}}{\arg\min} \sum_{i}^{N_c} \mathscr{L}_{\text{match}}(x_i, \hat{x}_{\rho(i)}) \tag{1}$$

where $\mathscr{L}_{\text{match}}(x_i, \hat{x}_{\rho(i)})$ is the pair-wise match cost between the prediction at index $\rho(i)$ and the ground truth tuple x_i.

3.1.2. Hungarian Loss

After matching, the model is trained to minimise the Hungarian loss. We denote the predicted patch as $\hat{\gamma}_{\rho(i)}$. Thus, the Hungarian loss is defined as in (2)

$$\begin{aligned}\mathscr{L}_{\text{hung}}(x_i, \hat{x}) = \sum_{i}^{N_c} [&\lambda_{\text{pose}} 1_{c_i \neq \varnothing} \mathscr{L}_{\text{pose}}(R_i, t_i, \hat{R}_{\hat{\rho}(i)}, \hat{t}_{\hat{\rho}(i)}) \\ &-\log \hat{P}_{\rho(i)}(c_i) + 1_{c_i \neq \varnothing} \mathscr{L}_{\text{patch}}(\gamma_i, \hat{\gamma}_{\hat{\rho}(i)})]\end{aligned} \tag{2}$$

where $\hat{\rho}$ is the lowest cost obtained by (1), c_i is the class probability and γ_i is a vector defining the ground truth image patch coordinates, height and width.

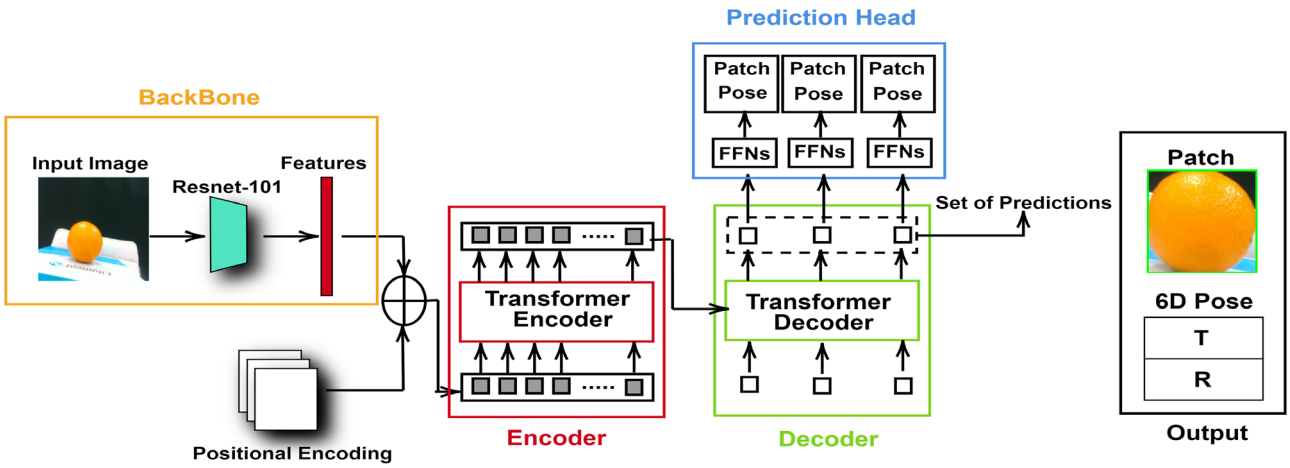

Figure 2. Transformer for detection, image patch, and initial 6D pose regression.

3.1.3. Patch Loss

The patch loss $\mathscr{L}_{\text{patch}}(\gamma_i, \hat{\gamma}_{\rho(i)})$ is a component in (2) and combines an l_1 norm loss and a generalized loss $\mathscr{L}_{\text{iou}}(\gamma_i, \hat{\gamma}_{\rho(i)})$,[51] as in (3)

$$\mathscr{L}_{\text{patch}}(\gamma_i, \hat{\gamma}_{\rho(i)}) = \sigma_1 \mathscr{L}_{\text{iou}}(\gamma_i, \hat{\gamma}_{\rho(i)}) + \sigma_2 ||\gamma_i - \hat{\gamma}_{\rho(i)}|| \tag{3}$$

and

$$\mathscr{L}_{\text{iou}}(\gamma_i, \hat{\gamma}_{\rho(i)}) = 1 - \left(\frac{|(\gamma_i \cap \hat{\gamma}_{\rho(i)}|}{|(\gamma_i \cup \hat{\gamma}_{\rho(i)}|} - \frac{|L(\gamma_i, \hat{\gamma}_{\rho(i)}) / \gamma_i \cup \hat{\gamma}_{\rho(i)}|}{|L(\gamma_i, \hat{\gamma}_{\rho(i)})|} \right) \tag{4}$$

where $\sigma_1, \sigma_2 \in \mathbb{R}$ are hyperprameters. $L(\gamma_i, \hat{\gamma}_{\rho(i)})$ is the largest patch having the ground truth γ_i and the predicted $\hat{\gamma}_{\rho(i)}$.

3.1.4. Pose Loss

$\mathscr{L}_{\text{pose}}(R_i, t_i, \hat{R}_{\hat{\rho}(i)}, \hat{t}_{\hat{\rho}(i)})$ is the pose loss. The pose loss is divided into two components, the translation t and the rotation R. Conventional l_2 norm loss is used to supervise the translation while a ShapeMatch loss L_{R},[5] is used for the rotation to deal with symmetrical objects.

$$\mathscr{L}_{\text{pose}}(R_i, t_i, \hat{R}_{\rho(i)}, \hat{t}_{\rho(i)}) = L_{\text{R}}(R_i, \hat{R}_{\rho(i)}) + ||t_i - \hat{t}_{\rho(i)}|| \tag{5}$$

$$L_{\text{R}} = \begin{cases} \frac{1}{|K|} \sum_{j_1 \in K} \min_{j_2 \in K} ||(R_i j_1 - \hat{R}_{\rho(i)} j_2)|| & \text{if symmetric} \\ \frac{1}{|K|} \sum_{j \in K} ||(R_i j - \hat{R}_{\rho(i)} j)|| & \text{otherwise} \end{cases} \tag{6}$$

where K represents the set of 3D points. R_i and t_i are the ground truth rotation and translation, respectively. $\hat{R}_{\rho(i)}$ and $\hat{t}_{\rho(i)}$ are the respective predicted object rotation and translation.

3.2. DEN

Depth estimation plays a crucial role in various applications,[52] including our implementation where the Depth Estimation Network (DEN) is tasked with predicting depth maps from monocular images. Inspired by the effectiveness of the Feature Pyramid Network (FPN)[22] in its ability to extract hierarchical features at multiple scales from input images and integration of features from different levels of abstraction, this functionality is essential in the estimation of depth, where objects and scene structures vary in size and complexity.

We employ a ResNet-101 backbone for feature extraction. This architecture leverages two successive 3×3 convolutional layers followed by ReLU activation functions, as illustrated in **Figure 3**. To enhance the up-sampling process for predicting finer details and covering a broader spatial context, we adopt the lightweight up-sampling technique proposed in ref. [53] This technique enables adaptive kernel generation, improving the accuracy of depth predictions by capturing contextual information effectively. The resulting depth images are scaled down to one-fourth of the original image size to balance computational efficiency and accuracy. Additionally, we compute the gradient of the depth map using a Sobel filter to further refine depth estimates.

The extracted depth information allows us to estimate additional translation parameters using the camera parameters. This method enables refinement of the pose estimation derived from our transformer model through weighting. The detailed formulation is given in Section 3.3.

The depth loss adopted in the training of our network is an l_1 norm loss defined as follows

$$\mathscr{L}_{\text{depth}} = \frac{1}{n} \sum_{i=1}^{n} ||d_i - \hat{d}_{(i)}|| \tag{7}$$

where, d_i and $\hat{d}_{(i)}$ are the ground truth depth and the predicted depth of every pixel i respectively.

3.3. Refinement Module for Final 6D Pose Estimation

The refinement module consists of the depth patch generation and final pose estimation processes. The patch and the regressed 6D pose from the transformer alongside the depth image are used as the inputs for the refinement module as shown in Figure 3.

The patch defined as the ROI obtained by the Detection and Regression Transformer is formulated as

$$\psi_i = [B_{\text{opx}}, B_{\text{opy}}, H_{\text{op}}, W_{\text{op}}] \tag{8}$$

where B_{opx}, B_{opy} represent the bottom left corner pixel coordinates of the patch and $H_{\text{op}}, W_{\text{op}}$ are the height and width of the patch respectively, all with respect to the original RGB image size (height and width) $S_{\text{o}} = (W_{\text{o}} \times H_{\text{o}})$. Similarly, let us represent the size of the depth image as $S_{\text{d}} = (W_{\text{d}} \times H_{\text{d}})$, where $S_{\text{o}} \neq S_{\text{d}}$. Depth patch ψ_j with respect to S_{d} can be obtained by (9).

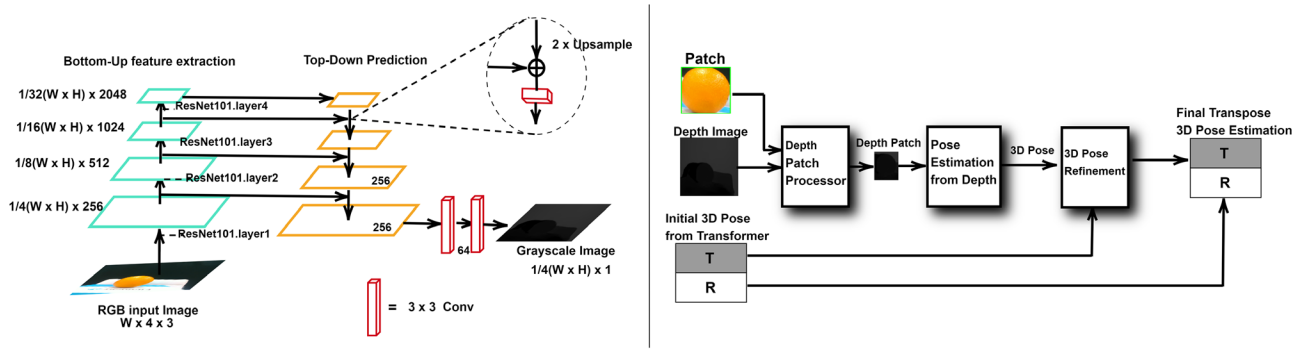

Figure 3. Left) Depth estimation network using feature pyramid, Right) Refinement module for final 3D pose estimation.

$$\psi_j = [B_{\mathrm{dpx}}, B_{\mathrm{dpy}}, H_{\mathrm{dp}}, W_{\mathrm{dp}}] = \psi_i \times \begin{bmatrix} \frac{W_{\mathrm{d}}}{W_{\mathrm{o}}} & 0 & 0 & 0 \\ 0 & \frac{H_{\mathrm{d}}}{H_{\mathrm{o}}} & c & 0 \\ 0 & 0 & \frac{H_{\mathrm{d}}}{H_{\mathrm{o}}} & 0 \\ 0 & 0 & 0 & \frac{W_{\mathrm{d}}}{W_{\mathrm{o}}} \end{bmatrix} \tag{9}$$

where B_{dpx},B_{dpy} now represents the bottom left pixel coordinates of the depth patch and H_{dp},W_{dp} are the height and width of the depth patch, respectively, all with respect to the depth image size S_{d}. The depth patch represents now our object ROI in the depth image frame and thus we can obtain the depth t_{z1} from the camera to the target as the depth information at the center pixel of the depth patch. The center pixel coordinates $C_{\mathrm{d}} = (C_{\mathrm{d}x}, C_{\mathrm{d}y})^{\mathrm{T}}$ can be obtained as follows

$$\begin{aligned} C_{\mathrm{d}x} &= B_{\mathrm{dp}x} + \frac{W_{\mathrm{d}p}}{2} \\ C_{\mathrm{d}y} &= B_{\mathrm{dp}y} + \frac{H_{\mathrm{d}p}}{2} \end{aligned} \tag{10}$$

The translation from the depth network model is $t_1 = (t_{x1}, t_{y1}, t_{z1})^{\mathrm{T}}$ where t_{z1} corresponds to the depth. Assuming the camera matrix is known, t_{x1} and t_{y1} can be obtained following the projection equation of a pinhole camera model as follows

Algorithm 1. TransPose for 6D object pose estimation.

Input: Monocular RGB image I
Output: 6D pose estimation (R,t)
Initialization:
Initialize ResNet-101 backbone, Transformer, FFNs, and DEN
Detection and Regression Transformer:
$F \leftarrow$ ExtractFeatures(I) ▷Extract feature vector F from image I using ResNet-101
$P \leftarrow$ TransformerEncoderDecoder(F) ▷Pass F through Transformer Encoder-Decoder to generate predictions P
For each prediction p in p **do**
  RegressPoseAndPatch(p) ▷Regress object pose (R,t) and image patch ψ using FFNs
  $L_{\mathrm{set\_pred}} \leftarrow$ SetPredictionLoss(p) ▷Compute set prediction loss $L_{\mathrm{set\_pred}}$
  $L_{\mathrm{hungarian}} \leftarrow$ HungarianLoss(p) ▷ Compute Hungarian loss $L_{\mathrm{hungarian}}$
  $L_{\mathrm{patch}} \leftarrow$ PatchLoss(p) ▷Compute patch loss L_{patch}
  $L_{\mathrm{pose}} \leftarrow$ PoseLoss(p) ▷Compute pose loss L_{pose}
End for
Depth Estimation Network (DEN):
$D \leftarrow$ ExtractDepthFeatures(I) ▷Extract depth features D from image I using ResNet-101
$D_{\mathrm{processed}} \leftarrow$ ProcessDepthFeatures(D) ▷Process D with convolutional layers and upsample using adaptive kernels
$L_{\mathrm{depth}} \leftarrow$ DepthLoss($D_{\mathrm{processed}}$) ▷Compute depth loss L_{depth}
Refinement Module for Final 6D Pose Estimation:
$\psi_{\mathrm{depth}} \leftarrow$ GenerateDepthPatch($D_{\mathrm{processed}}$) ▷Generate depth patch ψ from processed depth features
$t_{\mathrm{depth}} \leftarrow$ DepthTranslation(ψ_{depth}) ▷Compute depth translation t_{depth}
TransformerTranslation(P) ▷Compute transformer translation $t_{\mathrm{transformer}}$
$t_{\mathrm{final}} \leftarrow$ FuseTranslations($t_{\mathrm{depth}}, t_{\mathrm{transformer}}$) ▷Fuse translations to obtain final translation t_{final}
Training:
$L_{\mathrm{total}} \leftarrow L_{\mathrm{set\_pred}} + L_{\mathrm{hungarian}} + L_{\mathrm{patch}} + L_{\mathrm{pose}} + L_{\mathrm{depth}}$ ▷Total loss function
OptimizeParameters(L_{total}) ▷Optimize model parameters to minimize L_{total}
Testing and Evaluation:
Load test dataset (KITTI, YCB-Video, Fruity dataset)
For each test image in test dataset **do**
  $(R_{\mathrm{est}}, t_{\mathrm{est}}) \leftarrow$ PerformPoseEstimation(test image) ▷Perform pose estimation using TransPose
  EvaluatePerformance($R_{\mathrm{est}}, t_{\mathrm{est}}$) ▷Evaluate performance using metrics (e.g., eABS, eSQ, eRMS, elogRMS, eADD, eADDS)
End for
Output:
Final 6D pose estimations for test images

$$\begin{bmatrix} C_{ox} \\ C_{oy} \end{bmatrix} = \begin{bmatrix} f_x \frac{t_{x1}}{t_{z1}} + PP_x \\ f_y \frac{t_{y1}}{t_{z1}} + PP_y \end{bmatrix} \tag{11}$$

where f_x and f_y represent the focal length of the camera, $(\mathrm{PP}_x, \mathrm{PP}_y)^{\mathrm{T}}$ is the principal point. $C_o = (C_{ox}, C_{oy})^{\mathrm{T}}$ is the object centroid, which can be obtained from the image patch similarly to (10) to be $(B_{opx} + \frac{W_{op}}{2}, B_{opy} + \frac{H_{op}}{2})^{\mathrm{T}}$ assuming the centroid coincides with the center of the patch. Thus, t_{x1} and t_{y1} can be calculated as

$$\begin{bmatrix} t_{x1} \\ t_{y1} \end{bmatrix} = \begin{bmatrix} \frac{(C_{ox} - PP_x)t_{z1}}{f_x} \\ \frac{(C_{oy} - PP_y)t_{z1}}{f_y} \end{bmatrix} \tag{12}$$

Thus a complete translation from the depth image t_1 is obtained as

$$t_1 = (t_{x1}, t_{y1}, t_{z1})^{\mathrm{T}} \tag{13}$$

Finally, we can obtain the final fusion-based object translation t as

$$t = (w_1 \times t_1) + (w_2 \times t_2) \tag{14}$$

where the weights $w_1, w_2 \geq 0$ and $w_1 + w_2 = 1$. t_1 is the computed translation from the depth in (12) and t_2 is the regressed translation from the transformer model. Note that w_1 and w_2 are selected depending on the performance of both the transformer and depth model; the model with a lower loss will have a higher w and vice-versa.

4. Experimental Section

In this section, we present all the experiments conducted to test the capability of TransPose. Datasets are adopted and a comparison is made between TransPose and solutions available in the literature.

4.1. Dataset

The popular KITTI dataset is used as a benchmarking dataset for the depth estimation network likewise, the popular YCB-Video dataset[5] being a benchmark for 6D pose estimation. The dataset has 133 936 images of 640 × 480 pixels resolution. Each image is accompanied with bounding box labels, depths, segmentation, and 6D object pose annotations. Similar to ref. [5], a test was carried out on 2949 keyframes from 12 scenes. Additionally, we sampled from the Fruity dataset[54] to validate this approach in the context of fruit picking application.

4.2. Evaluation Metrics

The metrics adopted to evaluate the depth estimation network are the e_{ABS}, e_{SQ}, e_{RMS}, and e_{logRMS}, as proposed in ref. [55], as follows

$$e_{\mathrm{ABS}} = \frac{1}{|T|} \sum_{i=1}^{T} \frac{|d_i - \hat{d}_i|}{\hat{d}_i} \tag{15}$$

Table 1. Depth estimation network comparison with other methods. Bold is best performance.

Method	Evaluation metric (lower is better)			
	KITTI Dataset			
	e_{ABS}	e_{SQ}	e_{RMS}	e_{logRMS}
Make3D[59]	0.280	3.012	8.734	0.361
Eigen et al.[55]	0.190	1.515	7.156	0.270
Liu et al.[60]	0.217	1.841	6.986	0.289
Kuznietsov et al.[58]	**0.113**	0.741	**4.621**	0.189
TransPose	**0.114**	**0.724**	**4.694**	**0.185**
	Custom Fruit Dataset			
Eigen et al.[55]	0.0885	1.3000	4.2440	0.2115
Liu et al.[60]	0.0755	1.0917	3.9290	0.1938
Kuznietsov et al.[58]	0.0499	0.5350	2.6907	0.1427
TransPose	**0.0434**	**0.5153**	**2.5013**	**0.1342**

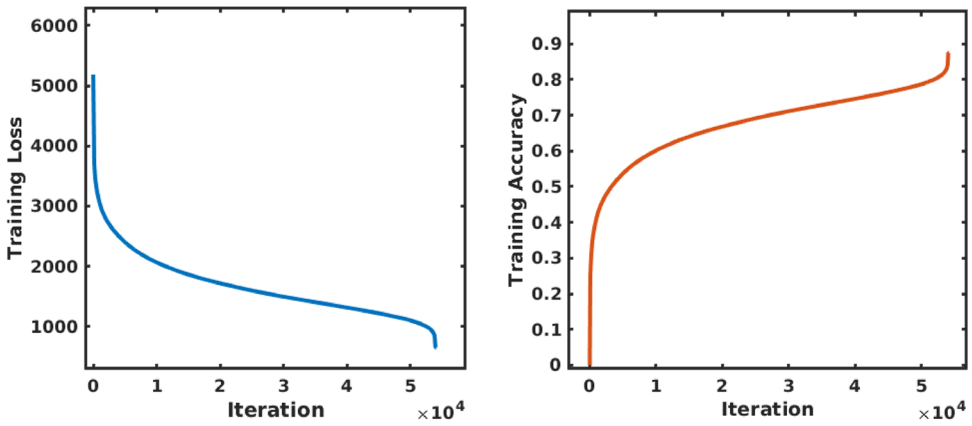

Figure 4. Training loss and accuracy per iteration.

$$e_{\text{SQ}} = \frac{1}{|T|}\sum_{i=1}^{T}\frac{||d_i - \hat{d}_i||^2}{\hat{d}_i} \tag{16}$$

$$e_{\text{RMS}} = \sqrt{\frac{1}{|T|}\sum_{i=1}||d_i - \hat{d}_i||^2} \tag{17}$$

$$e_{\text{logRMS}} = \sqrt{\frac{1}{|T|}\sum_{i=1}||\log d_i - \log \hat{d}_i||^2} \tag{18}$$

where T is the number of pixels in the test set.

(15) quantifies the average percentage error in depth estimation. It measures the absolute difference between the predicted depth $\hat{d}_i$ and the ground truth depth d, normalized by the predicted depth $\hat{d}_i$. (16) computes the average squared difference. This metric penalizes larger errors more severely than e_{SQ} due to the squared term. (17) measures the typical deviation of the predicted depths from the ground truth depths. (18) measures the root mean squared error of the logarithms of predicted depth and the ground truth. It is useful for the range of depth values that are large.

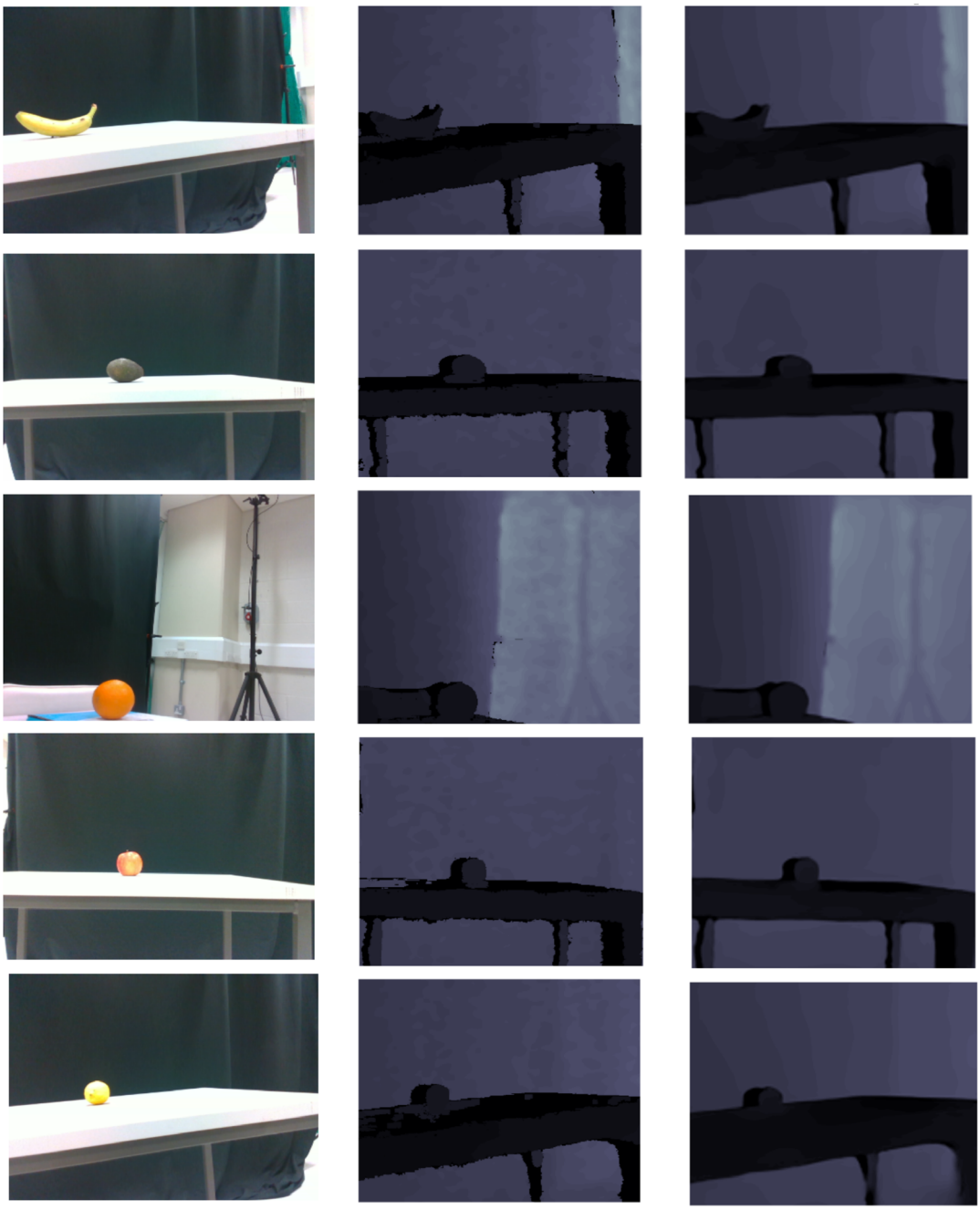

Figure 5. Qualitative results of the depth prediction network. The left column shows the ground truth RGB Images of five fruit classes. The middle column shows the ground truth depth images of the corresponding RGB images. The right column shows the predicted depth images from our network.

For the evaluation of the overall pose estimation, the average distance of descriptor (ADD) metric, as suggested in ref. [56], is used. This metric calculates the mean pairwise distance as follows

$$e_{\text{ADD}} = \frac{1}{|K|}\sum_{j\in K}||(R_j + t) - (\hat{R}_j + \hat{t})|| \tag{19}$$

where R and t are the ground truth object rotation and translation, respectively. $\hat{R}$ and $\hat{t}$ are the predicted rotation and translation respectively. K is the set of 3D points.

ADD is calculated as the closest point distance for symmetrical objects as follows

$$e_{\text{ADDS}} = \frac{1}{|K|}\sum_{j_1\in K}\min_{j_2\in K}||(R_{j_1} + t) - (\hat{R}_{j_2} + \hat{t})|| \tag{20}$$

(19) evaluates the accuracy of 6D pose estimation by computing the mean Euclidean distance between corresponding 3D points of the predicted pose and the ground truth pose. (20) calculates the average distance between the ground truth pose and the predicted pose using the minimum distance between the corresponding points. It is useful for symmetrical objects where there is ambiguity in choosing corresponding points the pseudo-code of the entire pipeline is enumerated in **Algorithm 1**.

4.3. Training

The model is initialized using pre-trained weights as described in ref. [13] It processes input images of size 640 × 480. The initial learning rate is set to 10^{-3} and decayed during training. A batch size of 16 samples is employed, with optimization performed using the AdamW optimizer.[57]

The hyper-parameters σ_1 and σ_2 for calculating $\mathscr{L}_{\text{patch}}$ in (3) are set to 2 and 5, respectively. Additionally, the parameter λ_{pose} for calculating $\mathscr{L}_{\text{hungarian}}$ in (2) is set to 0.05. The number of prediction queries N_c is fixed at 21.

4.4. Results

4.4.1. Depth Estimation Results

For the depth estimation network, the training loss and accuracy per iteration are shown in **Figure 4**. As the training proceeds, the training loss decreases thereby increasing the training accuracy per iteration.

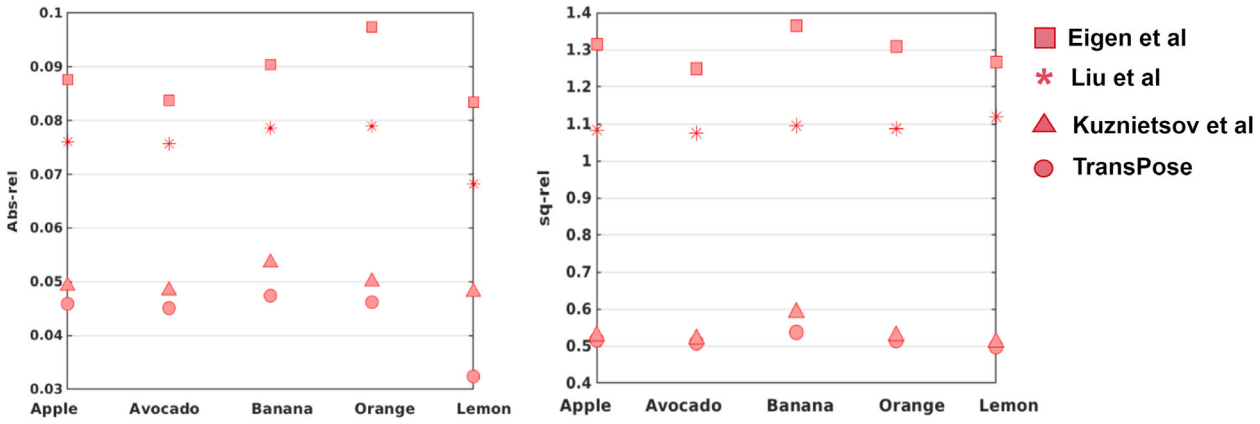

Figure 6. Comparison of TransPose with other methods in the literature on fruit classes in terms of e_{ABS} and e_{SQ} metrics.

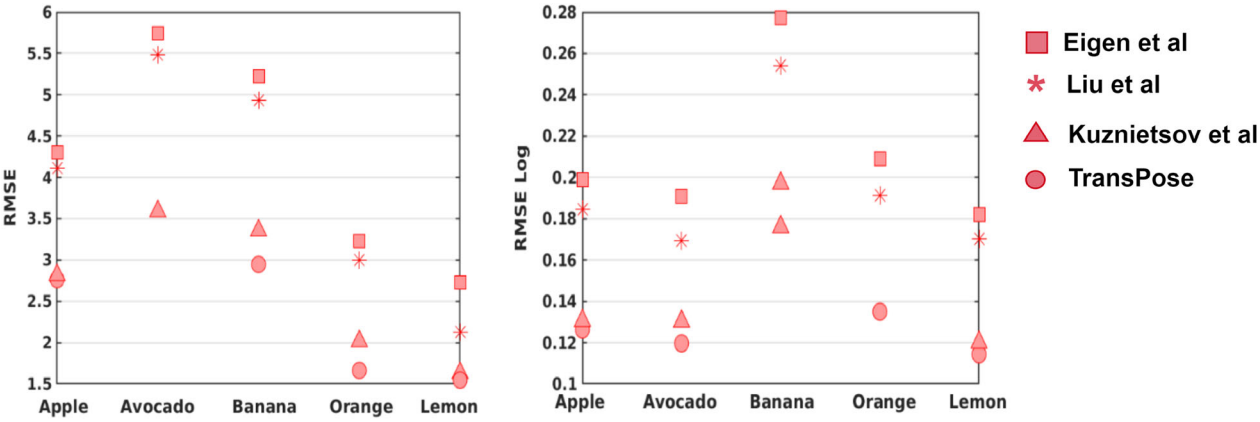

Figure 7. Comparison of TransPose with other methods in the literature on fruit classes in terms of e_{RMS} and e_{logRMS}.

The results obtained for the depth evaluation using the metrics in (15)–(18) are presented in **Table 1**.

We compare the performance of our depth estimation network with other methods on the popular KITTI dataset and our custom fruit dataset. On the KITTI dataset, our method outperformed the others in the e_{SQ} and e_{logRMS} metrics and compares very closely with[58] in the e_{ABS} and e_{RMS} metrics. On the fruit dataset, our network outperforms the other in e_{ABS}, e_{SQ}, e_{logRMS} metrics and compares closely in the e_{RMS} metric. This comparison shows the accuracy of our network as compared with others available in the literature and the flexibility to adapt for depth estimation as part of the TransPose pipeline. It is worth noting that higher depth accuracy comes at a computational cost and the depth estimation network is just one part of the TransPose pipeline. Thus, a reasonable trade-off between computational cost and accuracy is established to satisfy both

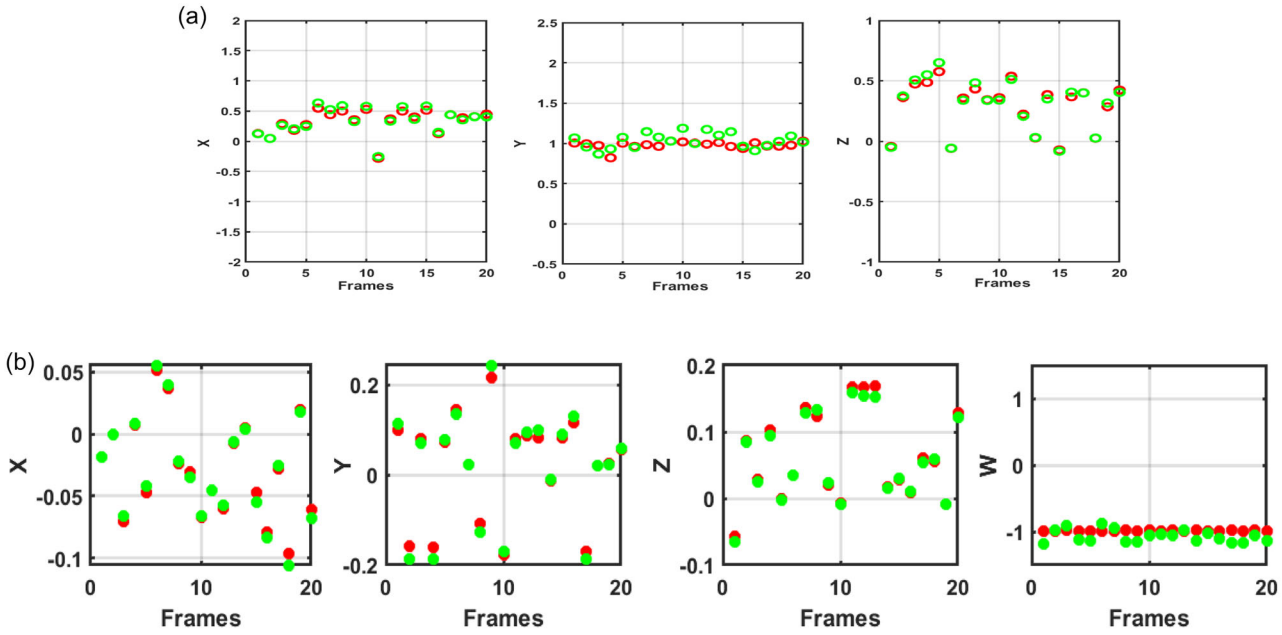

Figure 8. Pose estimate performance of TransPose for apple fruit class. Red is the ground truth while green is the prediction. a) Translation $[t_x, t_y, t_z]^T$ across 20 frames. b) Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ across 20 frames.

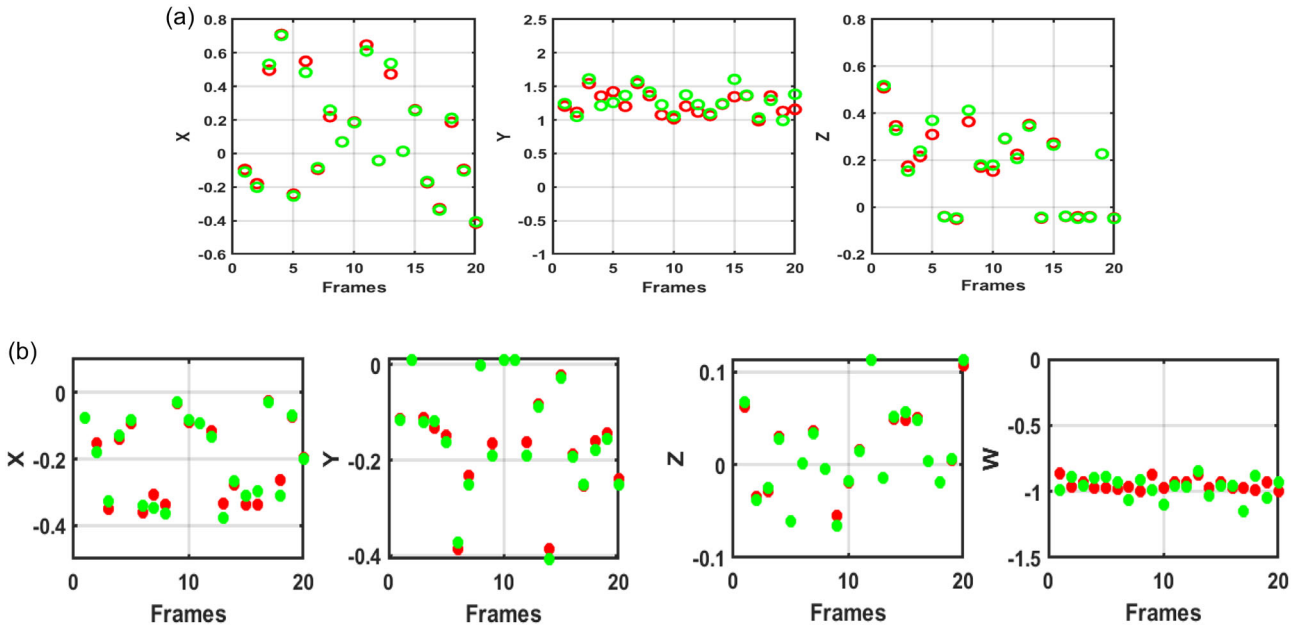

Figure 9. Pose estimate performance of TransPose for avocado fruit class. Red is the ground truth while green is the prediction. a) Translation $[t_x, t_y, t_z]^T$ across 20 frames. b) Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ across 20 frames.

decent estimation and future real-time implementation. Hence, the depth results are considered satisfactory.

The depth estimation qualitative results are shown in **Figure 5**. Samples from all the classes of our Fruit dataset including their ground truths and the corresponding predictions are shown. A colour map is added to the depth images for better visualization and evaluation.

Further comparisons with other methods are carried out across each individual class of fruit. **Figure 6** shows the comparison of each class of the fruit dataset using the e_{ABS} and e_{SQ} metrics. From the results, it is observed that TransPose outperformed all the methods across all the fruit classes. For the e_{SQ}, TransPose performs better in the banana class and slightly performs better in the other fruit classes.

Figure 7 compares the e_{RMS} and e_{logRMS} of each class of the fruit dataset. TransPose performs better on the banana, orange and lemon classes in terms of e_{RMS} metric and compares with[58] on the apple and avocado classes. For the e_{logRMS} metric, TransPose outperforms in the apple, avocado, banana, and lemon classes.

4.4.2. Performance of TransPose in Pose Estimation

We sample 20 test frames for the 6D pose estimation and compare the ground truth and the predicted poses. The translation

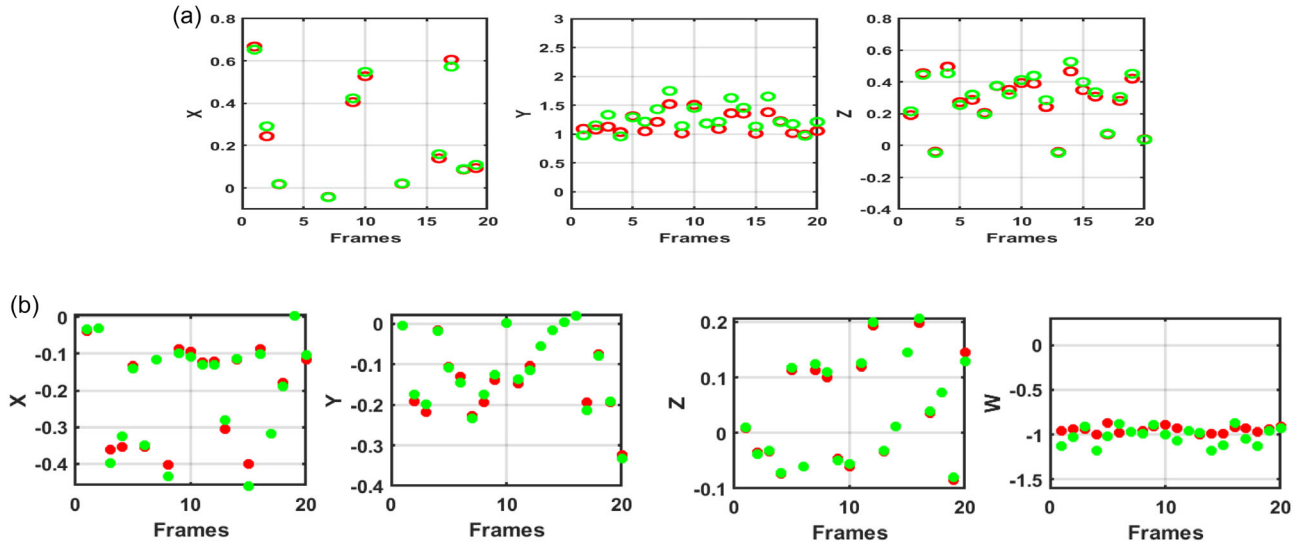

Figure 10. Pose estimate performance of TransPose for banana fruit class. Red is the ground truth while green is the prediction. a) Translation $[t_x, t_y, t_z]^T$ across 20 frames. b) Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ across 20 frames. a) Translation $[t_x, t_y, t_z]^T$ across 20 frames. b) Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ across 20 frames.

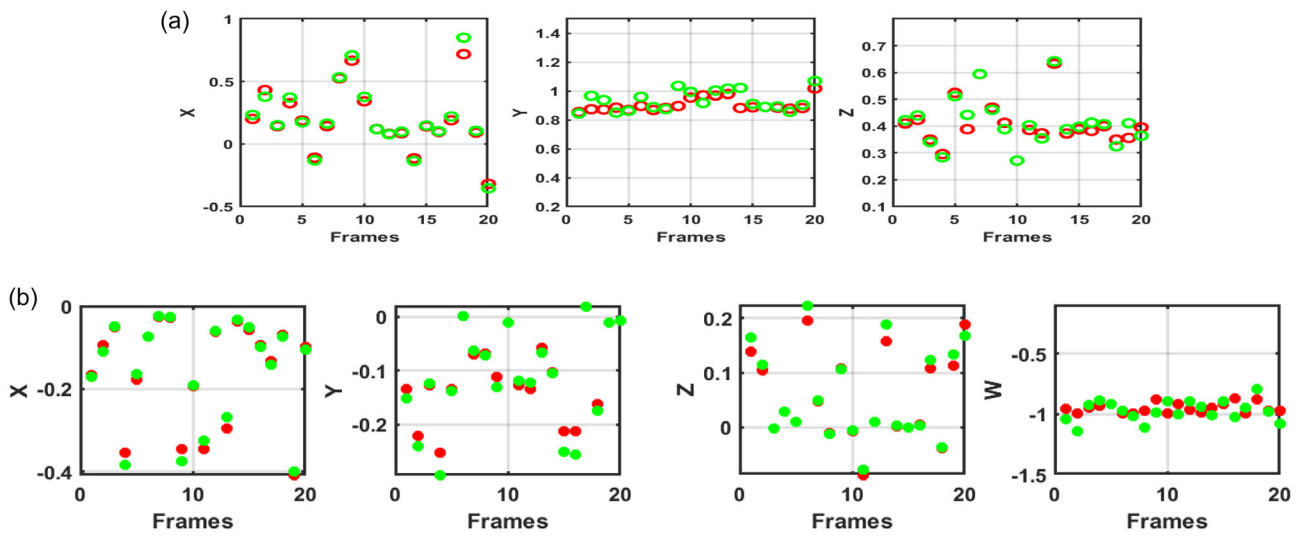

Figure 11. Pose estimate performance of TransPose for lemon fruit class. Red is the ground truth while green is the prediction. a) Translation $[t_x, t_y, t_z]^T$ across 20 frames. b) Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ across 20 frames.

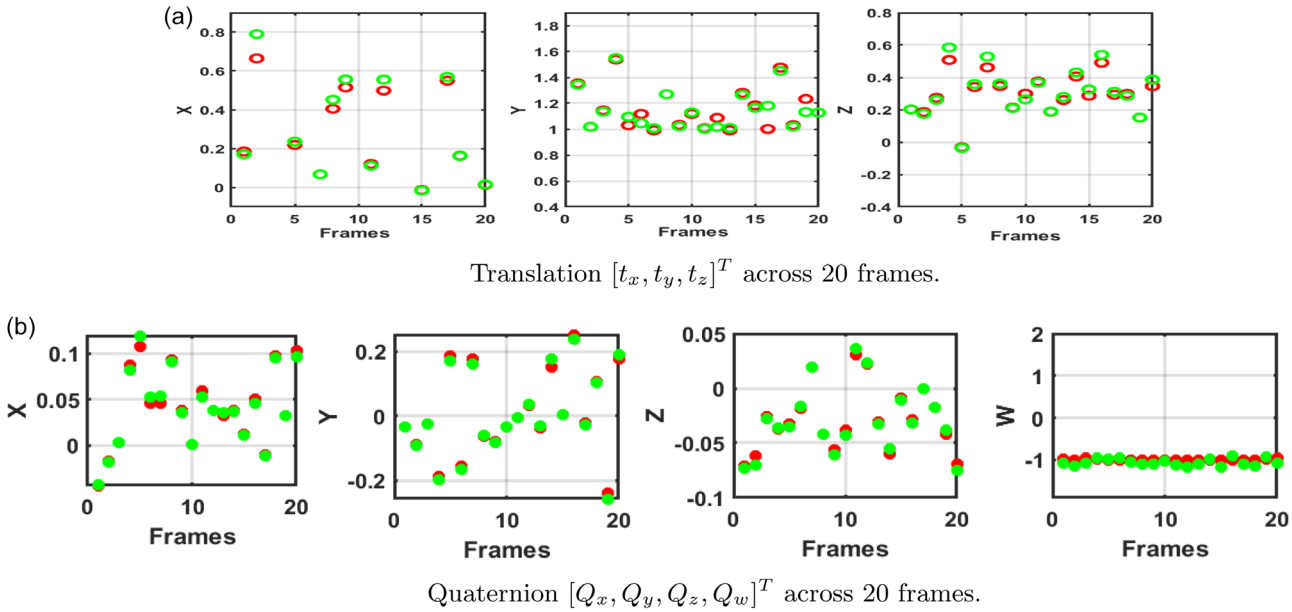

Figure 12. Pose estimate performance of TransPose for orange fruit class. Red is the ground truth while green is the prediction.

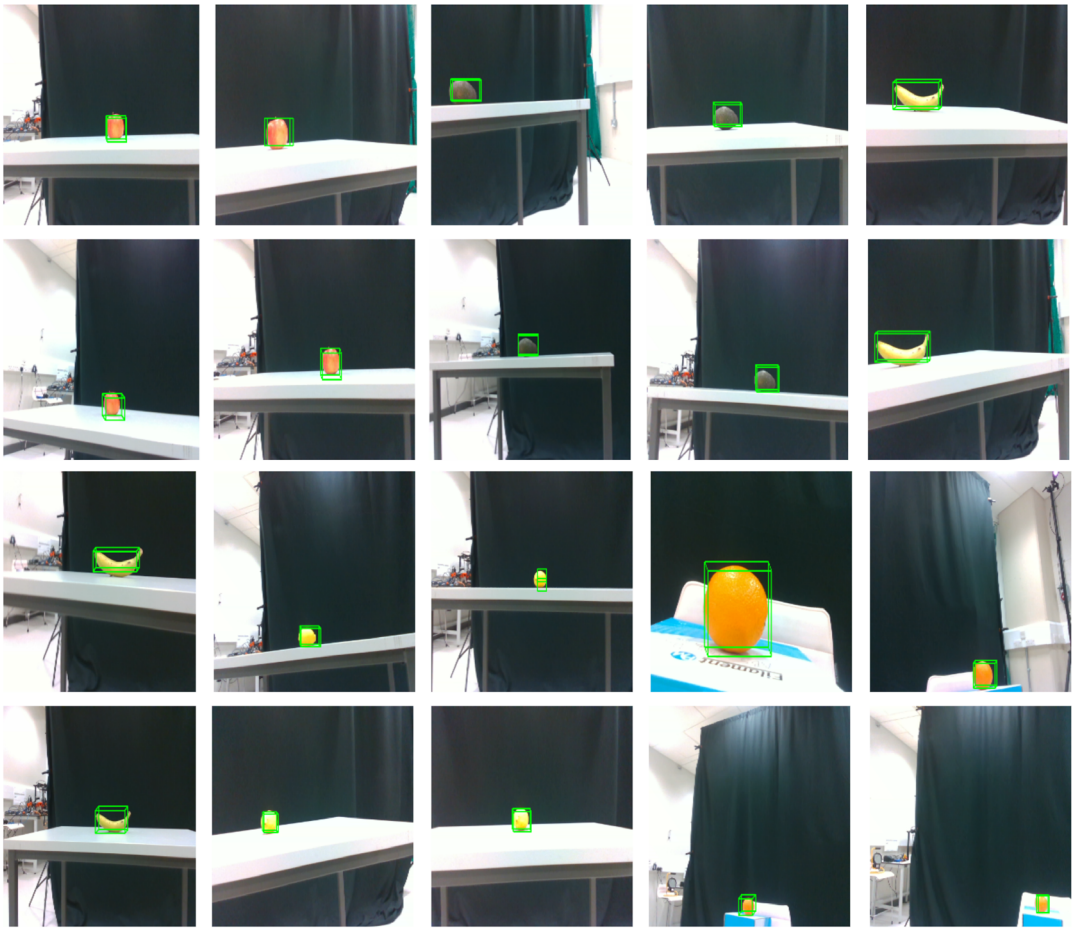

Figure 13. Qualitative samples from the fruit dataset across all the classes.

$[t_x, t_y, t_z]^T$ and the Quaternion $[Q_x, Q_y, Q_z, Q_w]^T$ which define the orientation are compared for all fruit classes as shown in **Figure 8–12**.

The samples are randomly selected from the test data to visualize the difference between the ground truth and the prediction. We can see that TransPose prediction solution matches well with the ground truth poses across all the fruit classes. The qualitative results we obtained for some sample frames from the fruit dataset are shown in **Figure 13**.

Table 2 shows a detailed evaluation of the objects from the YCB dataset using the e_{ADD} and e_{ADDS} metrics.

Table 2 demonstrates that TransPose outperforms the other methods considering the ADD metric for all the objects except the "tuna fish can", "bowl", "wood block", and "banana" where it closely compares with the other methods. Similarly, using the ADD-S metric, TransPose outperforms the other methods except for the objects "tuna fish can" and "wood block".

A similar comparison is conducted for the newly acquired fruit dataset using the ADD and ADD-S metric as shown in **Table 3**.

Table 2 and 3 show the overall performance of TransPose across the sample objects. From the average e_{ADD} and e_{ADDS} values, we can see that the depth refinement module improves the performance of 6D pose estimation.

Table 2. Pose estimation performance comparison using various approaches on some objects from YCB-V dataset. Symmetrical objects are highlighted in italics.

e_{ADD} (Higher is better)			
Object	T6D-Direct	PoseCNN	TransPose
Mug	72.1	57.7	**75.7**
Tuna fish can	59.0	**70.4**	60.2
Sugar box	81.8	68.6	**84.5**
Bowl	**91.6**	69.7	89.7
Master chef can	61.5	50.9	**63.4**
Tomato soup can	72.0	66.0	**75.6**
Wood block	**90.7**	65.8	**90.7**
Pudding box	72.7	62.9	**78.3**
Banana	87.4	**91.3**	90.4
Bleach cleanser	65.0	50.5	**70.2**
e_{ADDS} (Higher is better)			
Mug	89.8	78.0	**90.1**
Tuna fish can	**92.2**	87.9	91.7
Sugar box	90.3	84.3	**93.1**
Bowl	91.6	69.7	**92.3**
Master chef can	91.9	84.0	**92.4**
Tomato soup can	88.9	80.9	**90.8**
Wood block	**90.7**	65.8	90.6
Pudding box	85.1	79.0	**88.1**
Banana	93.8	85.9	**94.5**
Bleach cleanser	83.0	71.9	**84.3**
Average (e_{ADD})	75.38	65.38	**77.87**
Average (e_{ADDS})	88.50	80.44	**91.52**

Table 3. Pose estimation performance comparison using various approaches on fruit Dataset.

e_{ADD} (Higher is better)			
Object	T6D-Direct	PoseCNN	TransPose
Apple	78.7	62.4	**82.4**
Avocado	81.3	71.4	**82.6**
Banana	90.4	76.6	**92.4**
Orange	71.4	59.7	**79.3**
Lemon	89.5	71.9	**89.8**
e_{ADDS} (Higher is better)			
Apple	87.5	73.2	**89.7**
Avocado	86.2	82.9	**92.6**
Banana	92.7	82.3	**93.2**
Orange	84.6	80.2	**87.8**
Lemon	91.5	83.6	**94.3**
Average (e_{ADD})	82.26	68.40	**85.30**
Average (e_{ADDS})	89.73	78.74	**90.79**

4.4.3. Inference Time

We implemented the algorithm on a computer system equipped with an NVIDIA GeForce RTX 2060 GPU and an Intel Core i7 2.60 GHz CPU. The average processing times for a single input image are as follows: 31 ms for depth estimation, 23 ms for the transformer component, and 2 ms for the refinement module. This results in an average total time of 56 ms for the complete pipeline.

5. Conclusion

This article introduces TransPose, an advanced transformer-based network for 6D pose estimation enhanced by a depth refinement module to elevate overall performance. Unlike existing multi-modal networks that rely on multiple sensors and data types, TransPose achieves 6D pose estimation using only RGB images, augmented by depth information from a dedicated depth estimation network. The transformer network directly predicts 6D poses and incorporates depth refinement to enhance accuracy. We benchmark our depth estimation results against existing methods using standard evaluation metrics, demonstrating competitive performance. TransPose is evaluated across multiple datasets for both depth estimation and final 6D object pose regression. We expand our experimental scope to include a fruit dataset, validating the efficacy of our approach in precision agriculture.

In terms of limitations, TransPose currently relies on a depth estimation network whose performance may vary with environmental conditions and scene complexity to improve the overall accuracy. Future research could explore methods to enhance robustness to varying lighting conditions and occlusions, which are common challenges in real-world scenarios. Additionally,

while our approach shows promising results in controlled environments, its scalability and performance in diverse real-world settings warrant further investigation. Although there's always a trade-off between speed and accuracy, methods optimizing the computational efficiency of our pipeline, especially with the fine-tuning of the number of queries N_c can be investigated.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Keywords

depth estimation, pose estimation, transformer

Received: February 7, 2024
Revised: July 8, 2024
Published online:

[1] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, K. Daniilidis, in *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Hong Kong, China, May **2014**, pp. 3936–3943.
[2] M. Menze, A. Geiger, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, Massachusetts, June **2015**, pp. 3061–3070.
[3] D. Rondao, N. Aouf, in *2018 AIAA Guidance, Navigation, and Control Conf.*, Kissimmee, Florida, United States, January **2018**, p. 2100.
[4] E. Marchand, H. Uchiyama, F. Spindler, *IEEE Trans. Visual Comput. Graphics* **2015**, *22*, 2633.
[5] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox (preprint) arXiv:1711.00199, Submitted: November 2017.
[6] D. G. Lowe, in *Proc. of the Seventh IEEE Int. Conf. on Computer Vision*, Vol. *2*, IEEE, Kerkyra, Greece, September **1999**, pp. 1150–1157.
[7] A. Collet, M. Martinez, S. S. Srinivasa, *Int. J. Rob. Res.* **2011**, *30*, 1284.
[8] F. Rothganger, S. Lazebnik, C. Schmid, J. Ponce, in *2003 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2003. Proceedings*, Vol. *2*, IEEE, Madison, Wisconsin, June **2003**, pp. II–272.
[9] L. Bo, X. Ren, D. Fox, *Int. J. Rob. Res.* **2014**, *33*, 581.
[10] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, N. Navab, in *Asian Conf. on Computer Vision*, Springer, Daejeon Korea, November **2012**, pp. 548–562.
[11] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, C. Rother, in *European Conf. on Computer Vision*, Springer, Kongresshaus Zürich, September **2014**, pp. 536–551.
[12] C. Wang, D. Xu, Y. Zhu, R. Martn-Martn, C. Lu, L. Fei-Fei, S. Savarese, in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June **2019**, pp. 3343–3352.
[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, in *European Conf. on Computer Vision*, Springer, Glasgow, Scotland, August **2020**, pp. 213–229.
[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby (preprint), arXiv:2010.11929, Submitted: October 2020.
[15] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, M. Shah, *ACM Comput. Surv.* **2022**, *54*, 1.
[16] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, in *Int. Conf. on Machine Learning*, PMLR, Virtual, July **2021**, pp. 10347–10357.
[17] A. Amini, A. S. Periyasamy, S. Behnke, in *DAGM German Conf. on Pattern Recognition*, Springer, Bonn, germany, September **2021**, pp. 530–544.
[18] T. G. Jantos, M. A. Hamdad, W. Granig, S. Weiss, J. Steinbrener, in *6th Annual Conf. on Robot Learning*, Virtual, October, **2022**.
[19] A. Beedu, H. Alamri, I. Essa (preprint) arXiv:2210.13540, Submitted: October 2022.
[20] K. He, X. Zhang, S. Ren, J. Sun, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June **2016**, pp. 770–778.
[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July **2017**, pp. 2117–2125.
[23] D. G. Lowe, *Int. Comput. Vision* **2004**, *60*, 91.
[24] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, *Comput. Vision Image Understanding* **2008**, *110*, 346.
[25] E. Miyake, T. Takubo, A. Ueno, in *2020 IEEE/SICE Int. Symp. on System Integration (SII)*, IEEE, Virual, Janunary **2020**, pp. 960–965.
[26] X. Zhang, Z. Jiang, H. Zhang, Q. Wei, *IEEE Trans. Aerosp. Electron. Syst.* **2018**, *54*, 2342.
[27] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, in *2011 Int. Conf. Computer Vision*, IEEE, Barcelona, Spain, November **2011**, pp. 2564–2571.
[28] M. Calonder, V. Lepetit, C. Strecha, P. Fua, in *Computer Vision–ECCV 2010: 11th European Conf. on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, Springer, Heraklion, Crete, Greece, September **2010**, pp. 778–792.
[29] Z. Guo, Z. Chai, C. Liu, Z. Xiong, in *2019 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, IEEE, Hong Kong, China, July **2019**, pp. 1–6.
[30] S. Akizuki, Y. Aoki, in *2018 Int. Workshop on Advanced Image Technology (IWAIT)*, IEEE, Hong Kong, China, January **2018**, pp. 1–3.
[31] H. Yu, Q. Fu, Z. Yang, L. Tan, W. Sun, M. Sun, *IEEE Sens. J.* **2018**, *19*, 2217.
[32] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, V. Lepetit, *IEEE Trans. Pattern Anal. Mach. Intell.* **2011**, *34*, 876.
[33] Z. Cao, Y. Sheikh, N. K. Banerjee, in *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)* IEEE, Stockholm, Sweden, May **2016**, pp. 2441–2448.
[34] S. Tulsiani, J. Malik, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, June **2015**, pp. 1510–1519.
[35] A. Doumanoglou, V. Balntas, R. Kouskouridas, T.-K. Kim (preprint) arXiv:1607.02257, Submitted: July 2016.
[36] P. Wohlhart, V. Lepetit, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, June **2015**, pp. 3109–3118.
[37] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, C. Rother, in *Proc. of the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, December **2015**, pp. 954–962.
[38] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, C. Rother, *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, June **2016**, pp. 3364–3372.
[39] M. Rad, V. Lepetit, in *Proc. of the IEEE Int. Conf. on Computer Vision*, Venice, Italy, October **2017**, pp. 3828–3836.
[40] S. Peng, Y. Liu, Q. Huang, X. Zhou, H. Bao, in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June **2019**, pp. 4561–4570.

[41] Y. Hu, J. Hugonot, P. Fua, M. Salzmann, in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June **2019**, pp. 3385–3394.
[42] A. Kendall, M. Grimes, R. Cipolla, in *Proc. of the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, December **2015**, pp. 2938–2946.
[43] H. Su, C. R. Qi, Y. Li, L. J. Guibas, in *Proc. of the IEEE Int. Conf. on Computer Vision*, Santiago, Chile, December **2015**, pp. 2686–2694.
[44] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, R. Triebel, in *Proc. of the European Conf. on Computer Vision (ECCV)*, Munich, Germany, September **2018**, pp. 699–715.
[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, in *Computer Vision–ECCV 2016: 14th European Conf., Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, Amsterdam, The Netherlands, October **2016**, pp. 21–37.
[46] B. Tekin, S. N. Sinha, P. Fua, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, June **2018**, pp. 292–301.
[47] J. Redmon, A. Farhadi, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July **2017**, pp. 7263–7271.
[48] G. Gao, M. Lauri, X. Hu, J. Zhang, S. Frintrop, in *2021 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Xi'an, China, May **2021**, pp. 11081–11087.
[49] G. Gao, M. Lauri, Y. Wang, X. Hu, J. Zhang, S. Frintrop, in *2020 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Paris, France, May **2020**, pp. 3643–3649.
[50] X. Liu, G. Wang, Y. Li, X. Ji, in *Computer Vision–ECCV 2022: 17th European Conf., Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, Springer, Tel Aviv, Israel, October **2022**, pp. 499–516.
[51] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June **2019**, pp. 658–666.
[52] O. Araar, N. Aouf, J. L. Vallejo Dietz, *Ind. Rob.* **2015**, *42*, 200.
[53] J. Wang, K. Chen, R. Xu, Z. Liu, C. C. Loy, D. Lin, in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision*, Seoul, South Korea, October **2019**, pp. 3007–3016.
[54] M. Abdulsalam, Z. Chekakta, N. Aouf, M. Hogan, in *2023 31st Mediterranean Conf. on Control and Automation (MED)*., Limassol, Cyprus, July **2023**, pp. 144–149.
[55] D. Eigen, C. Puhrsch, R. Fergus, *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
[56] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, K. Daniilidis, in *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*, IEEE, Singapore, May **2017**, pp. 2011–2018.
[57] I. Loshchilov, F. Hutter (Preprint) arXiv:1711.05101, Submitted: Novermber 2017.
[58] Y. Kuznietsov, J. Stuckler, B. Leibe, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, July **2017**, pp. 6647–6655.
[59] A. Saxena, M. Sun, A. Y. Ng, *IEEE Trans. Pattern Anal. Mach. Intell.* **2008**, *31*, 824.
[60] F. Liu, C. Shen, G. Lin, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, MA, USA, June **2015**, pp. 5162–5170.