HOW TO TRACK YOUR DRAGON: A MULTI-ATTENTIONAL FRAMEWORK FOR REAL-TIME RGB-D 6DOF OBJECT POSE TRACKING

Isidoros Marougkas [1], *Petros Koutras* [1], *Nikos Kardaris* [1],
Georgios Retsinas [1], *Georgia Chalvatzaki* [2], *Petros Maragos* [1]

[1]School of E.C.E., National Technical University of Athens, 15773, Athens, Greece
[2]Department of Computer Science TU Darmstadt, 64289, Darmstadt, Germany
Email: {ismarougkas,nick.kardaris}@gmail.com, {pkoutras, maragos}@cs.ntua.gr, gretsinas@central.ntua.gr, georgia@robot-learning.de

ABSTRACT

We present a novel multi-attentional convolutional architecture to tackle the problem of real-time RGB-D 6D object pose tracking of single, known objects. Such a problem poses multiple challenges originating both from the objects' nature and their interaction with their environment, which previous approaches have failed to fully address. The proposed framework encapsulates methods for background clutter and occlusion handling by integrating multiple parallel soft spatial attention modules into a multitask Convolutional Neural Network (CNN) architecture. Moreover, we consider the special geometrical properties of both the object's 3D model and the pose space, and we use a more sophisticated approach for data augmentation for training. The provided experimental results confirm the effectiveness of the proposed multi-attentional architecture, as it improves the State-of-the-Art (SoA) tracking performance by an average score of 40.5% for translation and 57.5% for rotation, when testing on the dataset presented in [1], the most complete dataset designed, up to date, for the problem of RGB-D object tracking.

Index Terms— Pose, Tracking, Attention, Geodesic, Multi-Task

1. INTRODUCTION

Robust, accurate and fast object pose estimation and tracking, i.e. estimation of the object's 3D position and orientation, has been a matter of intense research for many years. The applications of such an estimation problem can be found in Robotics, Autonomous Navigation, Augmented Reality, etc. Although the Computer Vision community has consistently studied the problem of object pose estimation and tracking for decades, the recent spread of affordable and reliable RGB-D sensors like Kinect, along with advances in Deep Learning (DL) and especially the use of CNNs as the new SoA image feature extractors, led to a new era of research and a re-examination of several problems, with general aim the generalization over different tasks. CNNs have achieved ground-breaking results in 2D problems like object classification, object detection and segmentation. Thus, it has been tempting to the research community to increasingly use them in the more challenging 3D tasks.

The innate challenges of object pose estimation from RGBD streams include background clutter, occlusions (both static, from other objects present in the scene, and dynamic, due to possible interactions with a human user), illumination variation, sensor noise, image blurring (due to fast movement) and appearance changes as the object viewpoint alters. Moreover, one should account for the pose ambiguity, which is a direct consequence of the object's own geometry, in possible symmetries, the challenges of proper parameter representation of rotations and the inevitable difficulties that an effort of forging a model faces, when extracting information about the 3D scene geometry from 2D-projected images.

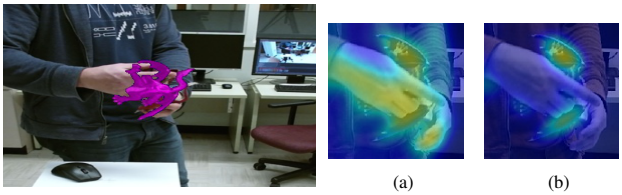

Fig. 1: The estimated object pose blended to each RGB frame along with the attention maps, which are learned by minimizing the two auxiliary binary cross entropy losses. The following tradeoff occurs: as the occlusion increases, foreground attention (a), which focuses on the moving parts of the scene (i.e. the hand and the object) gets blurrier, while occlusion attention (b) gets sharper and shifts focus from the object center to its body parts.

Previous works attempted to tackle the problem using DL, focusing on two different directions. The first family of proposed approaches in literature processes each video frame separately, without any feedback from the previous timeframe estimation. In [2], Xiang et al. constructed a CNN architecture that estimates binary object masks and then predicts the object class and its translation and rotation separately, while in [3] Kehl et al. extended the Single Shot Detection (SSD) framework [4] for 2D Object detection by performing discrete viewpoint classification for known objects. Finally, they refined their initial estimations via ICP [5] iterations. In [6] a CNN framework was proposed using RGB images for pixel-wise object semantic segmentation in a mask-level. Following this, UV texture maps are estimated to extract dense correspondences between 2D images and 3D object models minimizing cross entropy losses. Those correspondences are used for pose estimation via P'n'P [7]. This estimation is, ultimatelly, inserted as a prior to a refinement CNN that outputs the final pose prediction. More recently, iPose [8] is one of the attempts whose philosophy is the closest to ours. Its authors segment binary masks with a pretrained MaskRCNN [9] to extract background clutter and occluders and, they map 2D pixels to dense 3D object coordinates, which, in turn, are used as input to a P'n'P geometric optimization. Our attention modules have the same effect, but are computationally cheaper than MaskRCNN, as they relax the requirement for hard segmentation. The second category under study is temporal tracking, where feedback is utilized, to allow for skipping steps without prior knowledge of the previous pose. Garon et al. [10, 1], formulated the tracking problem exclusively as a learning one, by generating two streams of synthetic RGBD frame pairs from independent viewpoints and regressing the pose using a CNN. Liao et al. initialized a similar CNN architecture using a

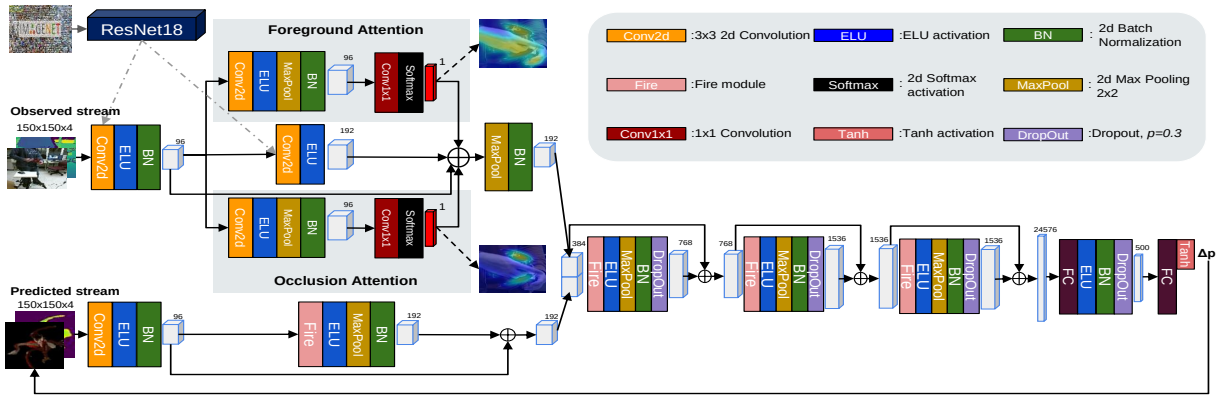

Fig. 2: Overview of the proposed CNN architecture for object pose tracking.

FlowNet2 [11] backbone and fused its two streams by subtraction. In [12], training was done with an Optical flow-based regularization term which encouraged the production of multiple heterogenous pose hypotheses that got bootstrapped in the final layer.

In this paper we extend the aforementioned approaches for object pose tracking, while building upon previous works [10, 1], delivering as main contributions:

- An explicit background clutter and occlusion handling mechanism that leverages spatial attentions and provides an intuitive understanding of the tracker's region of interest at each frame, while boosting its performance. To the best of our knowledge, this is the first such strategy, that explicitly handles these two challenges, is incorporated into a CNN-based architecture, while achieving real-time performance. Supervision for this mechanism is extracted by fully exploiting the synthetic nature of our training data.
- The use of a novel multi-task pose tracking loss function, that respects the geometry of both the object's 3D model and the pose space and boosts the tracking performance by optimizing auxiliary tasks along with the principal one.
- SoA real-time performance in the hardest scenario of the benchmark dataset [1], while achieving lower translation and rotation errors by an average of 40.5% for translation and 57.5% for rotation.

Accordingly, we provide the necessary methodological design details and experimental results that justify the importance of the proposed method in the challenging object pose tracking problem.

2. METHODOLOGY

2.1. Problem Formulation

Our problem consists in estimating the object pose $\mathbb{P}$, which is usually described as a rigid 3D transformation w.r.t. a fixed coordinate frame, namely an element of the Special Euclidean Lie group in 3D: $SE(3)$. It can be disentangled into two components; a rotation matrix R, which is an element of the Lie Group SO(3) and a translation vector $\mathbf{t} \in \mathbb{R}^3$. However, Bregier et al. [13] proposed a broader definition for the object pose, which can be considered as a family of rigid transformations, accounting for the ambiguity caused by possible rotational symmetry, noted as G $\in$ SO(3). We leverage this augmented mathematical definition for introducing a relaxation to the pose space $\mathscr{C}$ definition:

$$\mathscr{C} = \left\{ \mathbb{P} \quad | \quad \mathbb{P} = \left[\begin{array}{c|c} R \cdot G & \mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array} \right], \mathbf{t} \in \mathbb{R}^3, R \in \text{SO(3)}, G \in \text{SO(3)} \right\}. \quad (1)$$

For example, as stated in [13], the description of the pose of an object with spherical symmetry requires just 3 numbers: $(t_{x,y,z})$, as G can be any instance of SO(3) with the imprinted shape of the object remaining the same. Obviously, for asymmetrical objects, G=$\mathbb{I}_3$.

2.2. Architecture Description

The proposed architecture is depicted in Fig.2. Our CNN inputs two RGBD frames of size 150×150 frames: I(t),$\hat{\text{I}}$(t) (with I(t) being the "Observed" and $\hat{\text{I}}$(t) the "Predicted" one) and regresses an output pose representation $\Delta\mathbf{p} \in \mathbb{R}^9$, with 3 parameters for translation ($\hat{t}_{x,y,z} \in [-1, 1]$) and 6 for rotation. The first two layers of the "Observed" stream are initialized with the weights of a ResNet18[14], pretrained on Imagenet [15], to narrow down the real-synthetic domain adaptation gap, as proposed in [16]. Since ImageNet contains only RGB images, we initialize the weights of the Depth input modality with the average of the weights corresponding to each of the three RGB channels. Contrary to [16], we find beneficial not to freeze those two layers during training. The reason is that we aim to track the pose of the single objects we train on and not to generalize to unseen ones. So, overfitting to that object's features helps the tracker to focus only on distinguishing the pose change. To the output of the second "Observed" layer, we apply spatial attention for foreground extraction and occlusion handling and we add their corresponding output feature maps with the one of the second layer, along with a Residual connection [17] from the first layer. As a next step, we fuse the two streams by concatenating their feature maps and pass this concatenated output through three sequential Fire modules [18], all connected with residual connections [14].

Background and Occlusion Handling: After our first "Observed" Fire layer, our model generates an attention weight map by using a convolutional layer dedicated to occlusion handling and foreground extraction respectively, followed by a 1×1 convolution that squeezes the feature map channels to a weight map (normalized by softmax). Our goal is to distil the soft foreground and occlusion segmentation masks from the hard binary ground-truth ones (that we keep from augmenting the object-centric image with random backgrounds and occluders) in order to have their estimations available during the tracker's inference. To this end, we add the two corresponding binary cross entropy losses to our overall loss function. We argue our design choice of using two attention modules, as after experimentation, we found that assigning a clear target to each of the two modules is more beneficial, rather than relying on a single attention layer to resolve both challenges (see Sect.3.3.1).

Overall Loss and rotation representation: From a mathematical standpoint, immediate regression of pose parameters [1] with an Eu-

clidean loss is suboptimal: while the translation component belongs to the Euclidean space, the rotation component lies on a non-linear manifold of SO(3). Thus, it is straightforward to model the rotation loss using a Geodesic metric [19, 20] on SO(3), i.e. the length, in radians, of the minimal path that connects two of its elements: $\Delta\hat{R}, \Delta R$ (see Eq.3). In order to minimize the rotation errors due to ambiguities caused by the parameterization choice, we employ the 6D continuous rotation representation that was introduced in [21]: $\Delta\mathbf{r} = (\Delta\mathbf{r_x}^T, \Delta\mathbf{r_y}^T)$, where $\Delta\mathbf{r_{x/y}} \in \mathbb{R}^3$. Given $\Delta\mathbf{r}$, the matrix $\Delta R = (\Delta\mathbf{R_x}, \Delta\mathbf{R_y}, \Delta\mathbf{R_z})^T$ is obtained by:

$$\begin{aligned} \Delta\mathbf{R_x} &= N(\Delta\mathbf{r_x}) \\ \Delta\mathbf{R_y} &= N[\Delta\mathbf{r_y} - (\Delta\mathbf{R_x^T} \cdot \mathbf{r_y}) \cdot \boldsymbol{\Delta}\mathbf{R_x})] \\ \Delta\mathbf{R_z} &= \Delta\mathbf{R_x} \times \Delta\mathbf{R_y} \end{aligned} \tag{2}$$

where $\Delta\mathbf{R_{x/y/z}} \in \mathbb{R}^3$, $N(\cdot) = \frac{(\cdot)}{\|(\cdot)\|}$ is the normalization function. Furthermore, as it has already been discussed in [13], each 3D rotation angle has a different visual imprint regarding each rotation axis. So, we multiply both rotation matrices with a diagonal Inertial Tensor Λ, calculated on the object model's weighted surface and with respect to its center mass, in order to assign a different weight to each rotational component. We note here that since we want that matrix product to still lie in SO(3), we perform a Gramm-Schmidt orthonormalization on the Inertial Tensor Λ before right-multiplying it with each rotation matrix. Finally, we weigh the translation and rotation losses using a pair of learnable weights $\mathbf{v} = [v_1, v_2]^T$ that are trained along with the rest of the network's parameters using a Gradient Descent-based optimization method, as proposed by [22].

Symmetric Object Handling: In the special case of symmetric objects, we disentangle the ambiguities inserted due to this property from the core of the rotation estimation. We regress a separate Euler angle triplet of symmetry-based parameters $\hat{\mathbf{g}} \in \mathbb{R}^3$ that is converted to a rotation matrix $\hat{G}$, which gets right-multiplied with $\Delta\hat{R}$ before being weighted by the parameters of $\Lambda_{(G.S.)}$. We used a cylindrical cookiejar model for the symmetric object case, the shape of which has only one axis of symmetry. Consequentially, we estimate a single symmetry parameter, that of the object-centric z-axis. Before the conversion, that parameter is passed through a tanh function and multiplied by π to constrain its values.

As a result, our overall tracking loss function is formulated as:

$$\begin{aligned} L_{Track}(\Delta\hat{\mathbb{P}}, \Delta\mathbb{P}) &= e^{(-v_1)} \cdot MSE[(\Delta\hat{\mathbf{t}}, \Delta\mathbf{t})] + v_1 + v_2 + \\ &+ e^{(-v_2)} \cdot arcos\left(\frac{\text{tr}\left((\Delta\hat{R} \cdot \hat{G} \cdot \Lambda_{(G.S.)})^T \cdot (\Delta R \cdot \Lambda_{(G.S.)})\right) - 1}{2}\right) \end{aligned} \tag{3}$$

Using a similar external multi-task learnable weighting scheme ($\mathbf{s} = [s_1, s_2, s_3]^T$) as in (3), we combine our primary learning task, the pose tracking, with the two auxiliary ones: clutter and occlusion handling:

$$\begin{aligned} Loss &= e^{(-s_1)} \cdot L_{Track} + e^{(-s_2)} \cdot L_{Unoccl} + \\ &e^{(-s_3)} \cdot L_{Foregr} + s_1 + s_2 + s_3 \end{aligned} \tag{4}$$

2.3. Data Generation and Augmentation

Following [10], for our network (Fig.2), we generate two synthetic RGBD pairs I(t),$\hat{\mathbf{I}}(t)$ and we modify the augmentation procedure of [10, 1] as follows: Firstly, we blend the object image with a background image, sampled from a subset of the SUN3D dataset [23]. We also mimic the procedure of [10, 1] in rendering a 3D hand model-occluder on the object frame with probability 60%. A twist we added, is preparing our network for cases of 100% occlusion, by completely covering the object by the occluder for 15% of the occluded subset. Note that both the foreground and unoccluded object binary masks are kept during both of these augmentation procedures. Hence, we can use them as ground truth segmentation signals for clutter extraction and occlusion handling in our auxiliary losses to supervise the corresponding spatial attention maps. We add to the "Observed" frame pair I(t): (i) Gaussian RGB noise, (ii) HSV noise, (iii) blurring (to simulate rapid object movement), (iv) depth downsampling and (v) probabilistic dropout of one of the modalities, all with same parameters as in [1]. With a probability of 50%, we change the image contrast, using parameters $\alpha \sim U(0, 3)$, $\beta \sim U(-50, 50)$ (where U$(\cdot)$ is a uniform distribution) and gamma correction $\gamma \sim U(0, 2)$ with probability 50%, to help generalize over cases of illumination differences between rendered and sensor generated images. Instead of modelling the noise added to the "Observed" Depth modality with an ad-hoc Gaussian distribution as in [1], we consider the specific properties of Kinect noise [24] and model it with a 3D Gaussian noise (depending on depth and the ground truth object pose), used for simulating the reality gap between synthetic and real images. The rest of the preprocessing follows [10].

3. EVALUATION AND RESULTS

3.1. Implementation Details

We use ELU activation functions,a minibatch size of 128, Dropout with probability 0.3, Adam optimizer with corrected weight decay [25] by a factor $1e^{-5}$, learning rate $1e^{-3}$ and a scheduler with warm restarts [25] every 10 epochs. All network weights (except those transferred from ResNet18 [14]) are initialized via a uniform K.He [17] scheme. Since the Geodesic distance suffers from multiple local minima, following [26], we first warm-up the weights, aiming to minimize the LogCosh loss function for 25 epochs. Then, we train until convergence, minimizing the loss (4). The average training time is 12 hours in a single GeForce 1080 Ti GPU.

3.2. Dataset and Metrics

We test our approach on the "hard interaction scenario" of [1], which is considered the most difficult. It comprises of free 3D object motion, along with arbitrary occlusions by the user's hand. Our assumption is that if our proposed method performs better in the most challenging scenario, it will behave at least equally well in every other scenario. As in [1], we initialize our tracker every 15 frames, and use the same evaluation metrics. Due to limited computational resources, we produced only 20.000 samples, whose variability covers the pose space sufficiently enough, both for the ablation study and the final experimentation.

3.3. Ablation Study

3.3.1. Hierarchy choices for the attention modules

Here, we justify the need for both attention modules of our architecture (Fig. 2). We build upon the network proposed by [1], and we firstly introduce a single convolutional attention map just for occlusion handling . Then, we explore the possibility for a seperate attentional weighting of the "Observed" feature map for foreground extraction, prior to the occlusion one, and, we, finally, leverage both in parallel and add their resulting maps altogether.

The comparison of **Table 1** establishes not only the need for both attentional modules in our design, but also that parallel modules are optimal. We can observe the effect of parallel connection in Fig.1 as both attentions present sharper peaks. We can, also, observe a visual tradeoff between the parallel attentions: while the object is not occluded (either in steady state or when moving), the module responsible for foreground extraction is highlighted more intensely

	Translational Error (mm)	Rotational Error (degrees)
Garon et al. [1]	34.38 ± 24.65	36.38 ± 36.31
Only occlusion	17.60 ± 10.74	37.10 ± 35.08
Hierarchical clutter & occlusion	14.99 ± 9.89	39.07 ± 33.22
Parallel clutter & occlusion	$\mathbf{14.35 \pm 10.21}$	$\mathbf{36.98 \pm 32.29}$

Table 1: Comparison of different attentional foreground/occlusion handling configurations added to the baseline architecture of Garon et al.[1].

	Rotational Error(degrees)
Garon et al. [1]	36.38 ± 36.31
Rotational MSE	46.55 ± 40.88
Geod.	37.69 ± 35.39
Geod.+[21]	14.90 ± 21.76
Geod.+[21]+$\Lambda_{(G.S.)}$	$\mathbf{9.99 \pm 13.76}$

Table 2: The evolution of the proposed rotation loss, on the baseline architecture of Garon et al. [1] (without our proposed attention modules).

than the occlusion one. As the object gets more and more covered by the user's hand, the focus gradually shifts to the module responsible for occlusion handling. Note that this is not an ability we explicitly train our network to obtain, but rather a side effect of our approach, which fits our intuitive understanding of cognitive visual tracking.

3.3.2. Contributions of the rotation Loss components

We demonstrate the value of every component included in our rotation loss (leaving symmetries temporarily out of study), by: (i) regressing only the rotational parameters with the baseline architecture of [1], (ii), replacing the MSE loss with the Geodesic one, (iii), replacing the rotation parameterization of [1] with the continuous one of (2), and, (iv) including the Inertial Tensor weighting of each rotational component. **Table 2** indicates the value that translation estimation brings to rotation estimation, as when the former's regression is excluded, the latter's performance decreases. Moreover, **Table 2** justifies our progressive design selections in formulating our rotation loss, as with the addition of each ambiguity modelling, the 3D rotationerror metric decreases, starting from $46.55^o \pm 40.88^o$ reaching $9.99^o \pm 13.76^o$.

3.4. Experimental Results

According to our ablation study, we proceed to merge our parallel attention modules with the Geodesic rotation loss of (3), along with the remaining elements of Sect.2. We evaluate our method on two objects of [1]: one asymmetrical with rich texture and complex shape (dragon) and one symmetrical with poor texture and simple 3D shape (cookiejar).

3.4.1. The Dragon model: the asymmetric case

Our approach reduces mean errors by about 40.5% for translation and 57.7% for rotation w.r.t. baseline [1]. When the object is not occluded, the tracker focuses mostly on its 3D center, implicitly realizing in this way that this is the main 3D point of tracking interest. When the user's hand occludes parts of the dragon, the attention shifts to its body parts of interest that stand out of the grip, like its neck, wings or tail (Fig.1). The effectiveness of our method is demonstrated by the fact that, while [1] keeps track only of the object's 3D position under extreme occlusions, our improved version extends this property to 3D rotations as well. Although more computationally intense, the speed of our CNN (40 frames/sec.) still lies within the boundaries of real-time performance set by [1].

3.4.2. The CookieJar model: the symmetric case

For the special case of rotoreflective symmetry, we also report our results (**Table 3**) without/with accounting for the object's symmetry axis in formulating our rotation loss. We improve the approach of Garon et al.[1] by 3.35% and 31% in translation and rotation, respectively, if we do not take the symmetry degree of freedom into account in the loss. When we disentangle the rotation estimation and symmetries we try two different configurations: (i) learning one symmetry parameter over all possible pose changes in the training set and (ii) regressing a different one per pose pair. It is obvious that in the first case the symmetry parameter does not improve the tracker's performance, while in the second one it reduces both metrics to 14.69 ± 9.06mm/ $15.00^o \pm 13.20^o$. This occurs as, in the second case, the minimization of the tracking loss w.r.t. the symmetry matrix $\hat{G}$ (see [13]) achieves to fully exploit the extra degree of rotational freedom, by relaxing the global-solution constraint of the first one and allow one solution per pose pair. Finally, we improve [1] by 6.9% and 38% for translation and rotation, respectively. Here, the differences between our method and the baseline are lower than the ones of the asymmetric case. The attentions' effect is less prominent here since the cookiejar model is of simpler, symmetric shape and poorer texture. This replaces the distinctive clues of the dragon case (e.g. tails and wings standing out) with ambiguities, denying the corresponding modules of the ability to easily identify the pose.

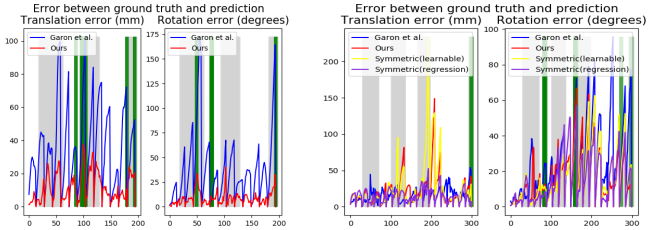

Fig. 3: The 3D translational and rotational error metrics, for the SoA(blue) and our(red), proposed approaches, for the dragon (on the left) and the cookiejar (on the right) models. In the symmetric case, the yellow curve describes learning a single symmetry parameter and the purple one regressing one per pose pair. Green intervals correspond to parts of the object movement where it is considered as fast, i.e. it exceeds a threshold of 12.5mm for inter-frame translation and 19^o for rotation. Gray intervals correspond to heavy occlusions.

	Translational Error(mm)	Rotational Error(degrees)	Tracking Fails
Garon et al. [1] (dragon)	34.38 ± 24.65	36.38 ± 36.31	17
Ours (dragon)	$\mathbf{11.63 \pm 8.79}$	$\mathbf{8.31 \pm 6.76}$	**2**
Garon et al. [1] (cookiejar)	15.78 ± 10.43	24.29 ± 20.84	15
Ours (cookiejar)	15.25 ± 16.06	16.73 ± 14.79	11
Ours with learnable symmetry parameter (cookiejar)	$\mathbf{19.9 \pm 19.51}$	$\mathbf{16.26 \pm 14.11}$	10
Ours with regression of symmetry parameter (cookiejar)	$\mathbf{14.69 \pm 9.06}$	$\mathbf{15.00 \pm 13.20}$	**9**

Table 3: Quantitative comparison of the results between the SoA [1] and our overall proposed method (with the modifications of residual connections, parallel attentions, Geodesic rotation loss and the sophisticated data augmentation w.r.t [1]) in terms of total errors and tracking fails. We consider a tracking failure to happen either when the translation error is more than 30cm or the rotation one more than 20^o for more than 7 consecutive frames.

4. CONCLUSION

In this work, we propose a CNN for fast and accurate single object pose tracking. We perform explicitly modular design of clutter and occlusion handling and we account for the geometrical properties of both the pose space and the object model during training. As a result, we reduce both SoA pose errors by an average of 40.5% for translation and 57.5% for rotation and gain an intuitive understanding of our artificial tracking mechanism. In the future, we aim to extend this work in the object-agnostic case and model temporal continuity of motion.

5. REFERENCES

[1] Mathieu Garon, Denis Laurendeau, and Jean-François Lalonde, "A framework for evaluating 6-dof object trackers," in *Proc. European Conf. on Computer Vision (ECCV)*, 2018, pp. 582–597.

[2] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[3] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 1521–1529.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *Proc. European Conf. on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

[5] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun, "Generalized-icp.," in *Robotics: science and systems*. Seattle, WA, 2009, vol. 2, p. 435.

[6] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic, "Dpod: Dense 6d pose object detector in rgb images," *arXiv preprint arXiv:1902.11020*, 2019.

[7] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua, "Epnp: An accurate o(n) solution to the pnp problem," *Int. J. of Comp. Vision (IJCV)*, vol. 81, 02 2009.

[8] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother, "ipose: instance-aware 6d pose estimation of partly occluded objects," in *Asian Conference on Computer Vision*. Springer, 2018, pp. 477–492.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask r-cnn," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[10] Mathieu Garon and Jean-François Lalonde, "Deep 6-dof tracking," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 11, pp. 2410–2418, 2017.

[11] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2462–2470.

[12] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox, "Deeptam: Deep tracking and mapping," in *Proc. European Conf. on Computer Vision (ECCV)*, 2018, pp. 822–838.

[13] Romain Brégier, Frédéric Devernay, Laetitia Leyrit, and James L Crowley, "Defining the pose of any 3d rigid object and an associated distance," *Int. J. of Comp. Vision (IJCV)*, vol. 126, no. 6, pp. 571–596, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Ieee, 2009, pp. 248–255.

[16] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proc. European Conf. on Computer Vision (ECCV)*, 2018, pp. 0–0.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[18] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size," *arXiv:1602.07360*, 2016.

[19] Du Q Huynh, "Metrics for 3d rotations: Comparison and analysis," *J. Math. Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[20] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li, "Rotation averaging," *Int. J. of Comp. Vision (IJCV)*, vol. 103, no. 3, pp. 267–305, 2013.

[21] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li, "On the continuity of rotation representations in neural networks," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5745–5753.

[22] Alex Kendall, Yarin Gal, and Roberto Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491.

[23] Jianxiong Xiao, Andrew Owens, and Antonio Torralba, "Sun3d: A database of big spaces reconstructed using sfm and object labels," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV)*, 2013, pp. 1625–1632.

[24] Chuong V Nguyen, Shahram Izadi, and David Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. IEEE, 2012, pp. 524–530.

[25] Ilya Loshchilov and Frank Hutter, "Fixing weight decay regularization in adam," *arXiv preprint arXiv:1711.05101*, 2017.

[26] Siddharth Mahendran, Haider Ali, and René Vidal, "3d pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2174–2182.