16720 Project Report: Rotation Representations in Deep Learning

Qiao Gu and Zhengyi(Zen) Luo

December 2019

1 Introduction

Many rotation representations have been developed in the history for different purposes and they have different properties, advantages, and applications. Recently, there have been more and more applications in computer vision and graphics that use deep neural networks to directly regress rotations. One example task is 6 degree of freedom object pose estimation. Although many related works solve 6dof pose by predicting correspondences, followed by RANSAC and PnP [1, 2, 3, 4, 5] which do not directly regress elements of rotation representations, some state-of-the-art works regress 4D unit quaternions as the final estimated rotations [6, 7] [7].

However, why they are using quaternion but not other representations and whether these representations are suitable for network regression have not attracted much attention in the deep learning community. As pointed out by a recent work [8], all of the popular 3D or 4D rotation representations (Euler angles, rotation vectors, quaternions) do not have a continuous mapping from rotation space to the representation space, which causes larger errors during direct regression at certain rotation angles. They proposed a 6-dimensional rotation representation (6DOF) based on the rotation matrix, which they proved to be continuous and performs better using direct regression. Further, they showed that all 3D and 4D rotation representations cannot achieve the continuous mapping from the rotation space to the representation space, and continuity requires at least a 5-dimensional rotation representation.

In this project, we reviewed several popular rotation representations, discussed their discontinuities and other problems in a direct regression task. Different from [8], we focused on giving an intuition on why those representations are discontinuous and their respective properties in a regression task. To test our hypothesis empirically, we generated a dataset based on ShapeNetV1 3D model dataset [9] and designed experiments on 3D rotation estimation task using CNN as model and images as input. We tried several traditional rotation representations as well as the 6DOF rotation representation proposed in [8] and we found that 3D and 4D rotation representations indeed yield larger training and testing errors due to their discontinuity, while rotation matrix works better and the 6DOF representation works best in the 3D rotation pose estimation task.

2 Theoretical Analysis

In this section, we will first analyze the desired properties of rotation representations in deep learning in Section.2.1 and then briefly review several mainstream rotation representations, including rotation matrix (Section. 2.2.1), Euler angle (Section. 2.2.2), rotation vector (Section. 2.2.3) and quaternion (Section. 2.2.4). For each rotation representation, we will analyze its discontinuity to see where the discontinuity lies and why this causes problems in deep neural networks that directly regress their elements.

Finally, in Section 2.2.5, we will briefly introduce the 6-dimensional rotation representation proposed in [8], and discuss its continuity and benefits in regression.

2.1 Continuity

Since neural network (or other machine learning model) regressors usually produce deterministic and continuous output, in tasks where a rotation representation is directly regressed, the output from an

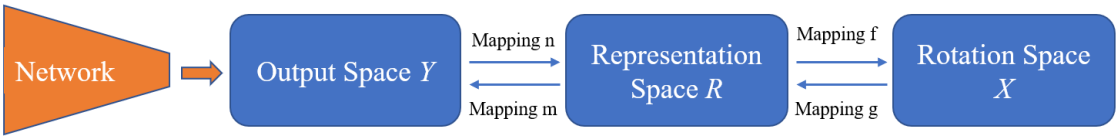


Figure 1: The overall relationships among output space, representation space and rotation space.

input of the same rotation should be in a continuous set. This means that if the input is perturbed by a very small shift, (e.g. rotated by a small angle) the output should only change a little within the corresponding set. Otherwise, the regressor will receive completely different and disjoint ground truth signal while the input signal is almost the same, and this will make the regressor converge slower or fail to converge.

Mathematically, as shown in Figure. 1, the original space X of 3D rotation is $SO(3)$. To avoid any ambiguity in the network training, the mapping g from X to representation space R must be continuous. Moreover, for some rotation representations, we need to normalize the direct output of the network to get valid representations as post-processing steps. For example, for rotation matrix, we need an orthogonalization step to ensure that the output matrices are in $SO(3)$, for Euler angles, we need to restrain its range in $[0, 2\pi]$, and for quaternions, the output needs to be normalized to get quaternions with unit norms. We call such mapping from the output space to the representation space n. Mapping n is normally a many-to-one mapping, but we can see that in order to eliminate any ambiguity in the network, the desired property is that a set in space Y that maps to the same representation in space R is connected and continuous. Otherwise, the network can produce the several disjoint sets of values for the input of the same rotation (like Euler angles of 0, 2π, 4π and two rotation vectors with opposite direction and complementary angles), and this is confusing to the neural networks in the training process.

In the following sections, we will see that all the 3- and 4- dimensional rotation representations are not continuous and cause problems in training regressors.

2.2 Rotation Representations Analysis

In this section, we will review popular rotation representations individually and analyze their properties, continuity, and whether they are suitable targets for regressors.

2.2.1 Rotation Matrix

A rotation matrix M is a matrix that can perform a rotation in Euclidean space. Rotation matrices in n-dimensional space form the special orthogonal group $SO(n)$, where $MM^T = M^TM = 1$ and $\det(M) = 1$.

In the 3D Euclidean space, a rotation matrix of the form

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} = \begin{bmatrix} | & | & | \\ \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \\ | & | & | \end{bmatrix} \tag{1}$$

can rotate any vector $v = [v_1,\ v_2,\ v_3]$ by left-multiplication, i.e. $M\mathbf{v}$, and such rotation can be viewed as rotating the original basis vector $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ to the three columen vectors $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ of M. The matrix-vector multiplication can be viewed as a weighted sum of the new basis vectors, which means $Mv = v_1\mathbf{m}_1 + v_2\mathbf{m}_2 + v_3\mathbf{m}_3$.

As a regression target, the rotation matrix has 9 degrees of freedom. The constraints given by $SO(3)$ prevent matrices from applying non-rigid transformation or reflection, but in the context of regression, such constraints cannot be easily imposed on the output. As suggested by [8], this needs to be achieved by a post-processing orthogonalization step, which tends to incur errors and slow down the convergence of deep regressors.

2.2.2 Euler Angle

Euler angles describe a series of 3 rotations along the major axes (x, y, and z axis) in the 3D space. The commonly adopted one is to rotate α around the z axis first, then β around the rotated x axis and γ around the rotated z axis, which is often denoted as $z - x - z$. By whether the rotation axes are fixed to the world coordinate frame or attached to and rotate with the object, the rotation indicated by Euler angles can be divided as intrinsic and extrinsic rotations.

Euler angles suffer from the problem of gimbal lock. For example, in the $z - x - z$ Euler angle, if the first and the third rotation axes coincide, then only their sum $\alpha + \gamma$ or their difference $\alpha - \gamma$ can be determined. These can causes ambiguity for neural networks that are supposed to yield deterministic output α, β and γ given a fixed input.

As a regression target, Euler angles have 3 degrees of freedom. While it does give an intuitive way of understanding rotation, its discontinuity is obvious: if we define the range of each Euler angle to be $[0, 2\pi]$, then rotating $\theta = 2\pi$ along an axis is the same rotation as $\theta = 0$. Imagine the situation where the network outputs 0.1 and the ground truth is $2\pi - 0.1$, which are quite close in the rotation space, but the loss to the network is quite large and the network needs to output two values far from each other. This will cause great confusion to the regressor and the error at such border case can be larger than normal.

2.2.3 Rotation Vector

Rotation vector is a representation for 3D rotations with its direction indicating the rotation axis and magnitude equal to the rotation angle. The space of rotation vector is a 3D ball of radius 2π, but since a rotation vector v represents the same rotation as another u if u and v have opposite orientation and $||u|| + ||v|| = 2\pi$, the whole 3D ball of rotation vector make rotation vector a double cover of the rotation space. This causes ambiguity to the neural network as they can output two opposite rotation vector for the same rotation. The ambiguity can be solved by constraining the space of rotation vectors to be a half-open ball [10] so that there exists a one-to-one mapping between rotation vectors and 3D rotation space.

As a regression target, Rotation Vector has 3 degrees of freedom. But the ambiguity discussed above incurs discontinuity at the edges and the open plane of the half-open ball, and similar to Euler angles, this continuity confuses the network and make output inconsistent. Also, a rotation vector encodes the angle of rotation along the axis as the vector norm, and such compact representation may be difficult for a regressor to learn.

2.2.4 Quaternion

Quaternion, as an extension of complex number system, has the form $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, where a, b, c and d are real numbers and $\mathbf{i}$, $\mathbf{j}$ and $\mathbf{k}$ are the quaternion units with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. With defining $\mathbf{v} = b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, we can also express a quaternion in the form of a scalar plus a vector as $a + \mathbf{v}$.

Unit quaternion is a quaternion with the unit norm, and can be used to represent a rotation or an orientation. The intuition behind is that if the rotation axis is $\mathbf{u}$ (a unit vector) and the rotation angle is α, we can define a quaternion

$$q = \cos\frac{\alpha}{2} + \mathbf{u}\sin\frac{\alpha}{2}, \tag{2}$$

and then the result of rotating a vector v along the axis $\mathbf{u}$ by α is

$$\mathbf{v}' = q\mathbf{v}q^{-1} = (\cos\frac{\alpha}{2} + \mathbf{u}\sin\frac{\alpha}{2})\mathbf{v}(\cos\frac{\alpha}{2} - \mathbf{u}\sin\frac{\alpha}{2}). \tag{3}$$

Quaternions have good properties when used to represent rotations. Compared to rotation matrices, quaternions are more compact, numerically stable and does not require orthogonalization process. Compared to Euler angles, quaternions do not suffer from gimbal lock and singularity.

However, we can easily see that quaternion also gives a double cover over the rotation space. Consider a rotation along vector $\mathbf{u}$ by α, and another along $-\mathbf{u}$ by $2\pi - \alpha$. Similar to rotation vectors, these are

essentially the same rotation but result in two different quaternions:

$$q = \cos\frac{\alpha}{2} + \mathbf{u}\sin\frac{\alpha}{2} \tag{4}$$

$$\begin{aligned} q' &= \cos(\frac{2\pi-\alpha}{2}) - \mathbf{u}\sin(\frac{2\pi-\alpha}{2}) \\ &= -\cos\frac{\alpha}{2} - \mathbf{u}\sin\frac{\alpha}{2} \\ &= -q. \end{aligned} \tag{5}$$

This ambiguity is usually solved by choosing the solution that has nonnegative a, but this breaks the continuity of the mapping from $SO(3)$ to the space of unit quaternions.

As a regression target, unit quaternion has 4 degrees of freedom. Normalization to unit length is usually enforced as a post-processing step and or enforced directly using softmax function.

2.2.5 6DOF Continuous Rotation Representation

In a recent work this year [8], a 6-dimensional rotation representation is proposed and proved to be continuously mapped from the rotation space $SO(3)$. Their idea is based on the rotation matrix. Given a rotation matrix in $SO(3)$

$$M = \begin{bmatrix} | & | & | \\ \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \\ | & | & | \end{bmatrix}, \tag{6}$$

the mapping from the rotation space X to the representation space is just simply taking the first 2 columns, and thus the rotation representation is a 6 dimensional regression target.

$$g\left(\begin{bmatrix} | & | & | \\ \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 \\ | & | & | \end{bmatrix}\right) = \begin{bmatrix} | & | \\ \mathbf{m}_1 & \mathbf{m}_2 \\ | & | \end{bmatrix} \tag{7}$$

To recover the rotation matrix from such 6 number representation, they used a Gram-Schmit like process and normalize and orthogonalize each column and get the last one by cross product. The mapping from representation space to the rotation space f is defined as follows:

$$\begin{aligned} f\left(\begin{bmatrix} | & | \\ \mathbf{a}_1 & \mathbf{a}_2 \\ | & | \end{bmatrix}\right) &= \begin{bmatrix} | & | & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 \\ | & | & | \end{bmatrix} \\ &= \begin{bmatrix} | & | & | \\ N(\mathbf{a}_1) & N(\mathbf{a}_2 - (\mathbf{b}_1 \cdot \mathbf{a}_2)\mathbf{b}_1) & \mathbf{b}_1 \times \mathbf{b}_2 \\ | & | & | \end{bmatrix}, \end{aligned} \tag{8}$$

where N is the normalization function that $N(\mathbf{a}) = \frac{\mathbf{a}}{||\mathbf{a}||}$.

This representation is essentially only letting network producing the first two columns of the rotation matrix and recover the third one in the post-processing stage. Using this rotation representation, the mapping g from the rotation space to the representation space is still one-to-many but continuous in the sense that all possible representations for a single rotation is a closed set and there are no representations corresponding to other rotation inside this set. On the contrary, the one-to-many mapping from rotation space to the Euler angle map the identity rotation to 0, 2π, 4π..., which is discretely spread over the space Y of network's possible output.

3 Experiments

In order to identify the best rotation representation for deep learning, we chose the task of 6 degrees of freedom (6dof) pose estimation as our evaluation task. Given an image, the goal of the task is to recover the rotation (3 degrees of freedom) and translation (3 degrees of freedom) information for an object in the scene. Some popular works in 6dof pose estimation usually use a convolutional neural network to process the visual data and directly regress pose information [7]. The task directly involves rotation

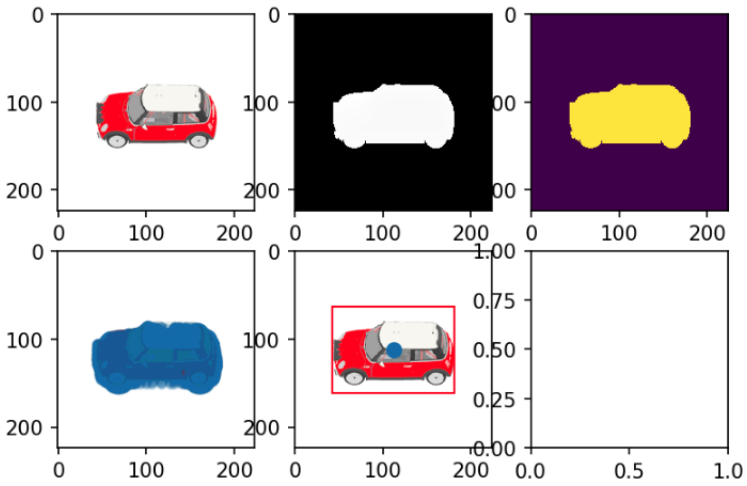


Figure 2: Generated data using ShapeNet 3D model. From left to right, top to down, each picture represents: RGB image, depth map, semantic segmentation, ground truth pose, 2D + 3D bounding box.

and the performance of the estimated pose (often evaluated in per point average distance between the ground truth pose and the estimated pose) reflects the quality of recovered rotation directly. To further disentangle irrelevant factors, our experiments assume known translation, so the problem reduces to a 3 degree of freedom rotation estimation problem.

3.1 Dataset

Popular pose estimation dataset such as YCB-video [6], LineMOD [11] usually involves multiple objects in a highly cluttered scene, and may introduce interfering factors such as camera parameters, lighting conditions, object appearances etc. that affect a neural network's performance. Thus, we generated a synthetic dataset in which the only varying factor is the object rotation. To generate this dataset, a random 3D model of an object (for instance, a car) is chosen from ShapeNetV1 3D model dataset [9]. The chosen model will be rotated randomly and rendered by the same camera parameters. We used the Blender for rendering and recovered ground truth rotation based on the current rotation configuration. We generated 10000 images with ground truth rotation and split them 4:1 into training and testing set. An example of the dataset can be seen in Figure 2.

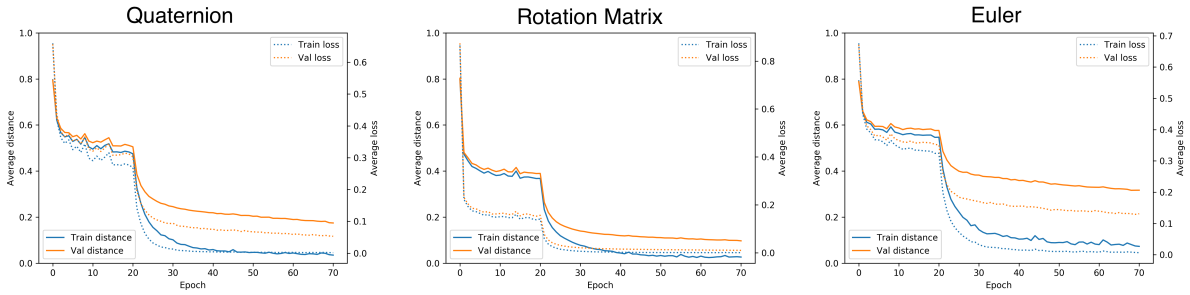
3.2 Network

To train a pose estimator, we use the Resnet-18 pretrained on ImageNet from PyTorch as feature extractor, re-initialize the fully connected layer based on the current representation's dimension, and directly output the rotation representation. Specifically, we have: quaternion (4d), rotation vector (3d), Euler angles (3d), rotation matrix (9d) and 6D representation (6d). To train our network, we first freeze up the previous layers of resnet18 and only finetune the last fully connected layers for 20 epochs, and then we unfreeze the whole network and train jointly for 50 epochs. To compute the loss, we first convert the rotation representation to rotation matrix (3x3) and use L2 loss on each of the elements from the rotation matrix. Notice that this loss is equivalent for all rotation representations, as described in [8]. We use a learning rate of 1e-3, and use the Adam optimizer from PyTorch.

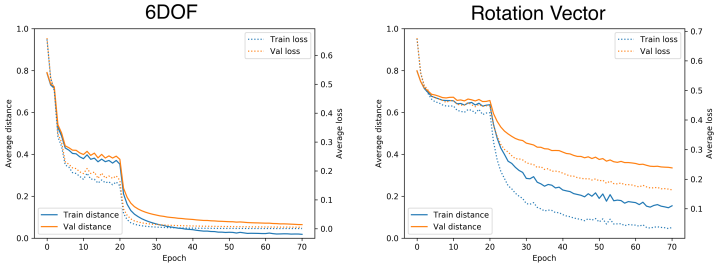
4 Results

4.1 Training and Testing Result

The training and testing plots can be found in Figure 3. To evaluate the network performance on recovering the rotation in a unified way, we calculate the average per point distance between ground



(a)



(b)

Figure 3: Plot of loss curve and projection distance

truth 3D model point cloud and 3D model rotated by the recovered rotation:

$$distance = \sum_{N}^{i=0} norm(R_{estimated} * \text{pos}_i - R_{gt} * \text{pos}_i)/N \tag{9}$$

In Figure 4 we report the average per point distance for the training and testing set for each rotation representation. Clearly, the 6D rotation representation performs the best in terms of loss and projection distance. The network also converges faster. Upon visual inspection, as shown in Figure 5, the recovered rotation is reasonable. Overall, the rotation vector (axis-angle) and Euler angles representation performed the worst, and quaternion does not generalize well from the train to test set. This is not surprising: rotation vector and Euler angles are both 3D representation and both suffer from discontinuity mentioned in the theory section. Quaternion is a 4-dimensional representation, but it still suffers from discontinuity. Rotation matrix has 9 degrees of freedom, and has the closest performance to the 6D representation. Nonetheless, the rotation matrix representation may suffer from redundancy, and the QR decomposition post-processing required to make the recovered rotation matrix orthonormal is not directly optimized through the network.

4.2 Failure cases

We can also observe a number of failure cases in the best performing representation (6DOF): Figure 6. In these cases, the ground truth rotation is along a fixed axis (45 degrees only the positive Y & Z plane). There can be a couple of reasons for model failure: it may be the case that these specific rotations are not prevalent in the training set, and model does not generalize well enough to cover these cases. It can also be possible that training a convolutional neural network to directly regress 3D rotation based on different viewpoints is a hard problem, and maybe better ways of interpreting or geometrical information (like 3D models) are needed.

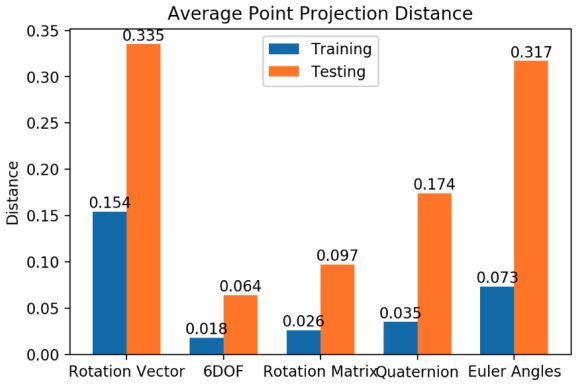


Figure 4: Average Point Projection distance

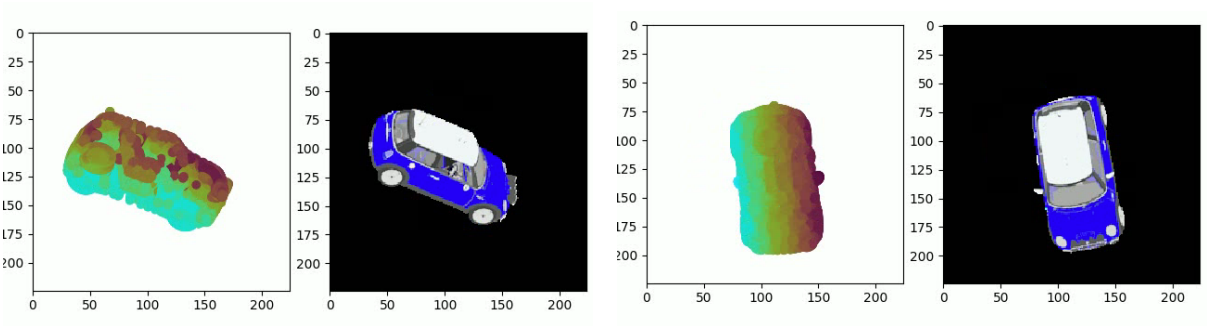


Figure 5: Visualization of recovered rotation (image from the test set). On the left is the projected point cloud using the recovered rotation, and on the right is the ground truth rendering.

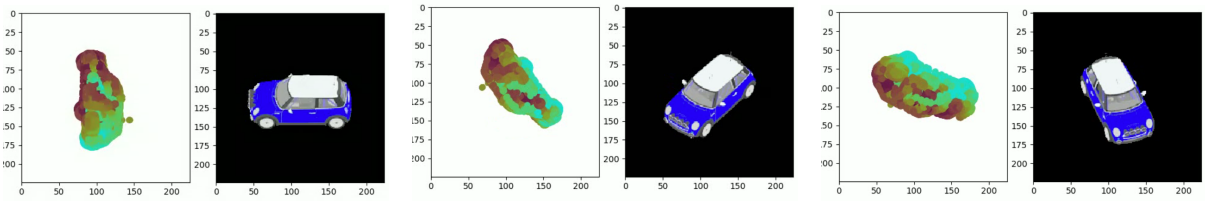


Figure 6: Visualization of failure cases.

5 Conclusion

In this report, we investigated the properties of popular rotation representations and tested our theoretical findings in the task of 6 dof pose estimation. From both the theoretical and experimental standpoints, popular 3D and 4D rotation representation such as quaternion, Euler angles, and rotation vectors suffer from discontinuity and does not perform as well as the newly developed 6D rotation representation discussed in [8]. The 6D rotation representation also does not suffer from the gimbal lock as Euler angles do. Thus, for applications using rotation representation and neural networks, the 6D rotation representation may be of interest and worth further exploration.

References

[1] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In David Fleet, Tomas

Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 536–551, Cham, 2014. Springer International Publishing.

[2] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[3] Kiru Park, Timothy Patten, and Markus Vincze. Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[4] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3848–3856, 2017.

[6] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. 2018.

[7] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. 2019.

[8] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[9] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[10] Carlo Tomasi. Vector representation of rotations. *Computer Science 527 Course Notes, Duke University*, 2013.

[11] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part I*, ACCV'12, pages 548–562, Berlin, Heidelberg, 2013. Springer-Verlag.