3D Object Pose Estimation using Multi-Objective Quaternion Learning

Christos Papaioannidis and Ioannis Pitas, *Fellow, IEEE*

***Abstract*—In this work, a framework is proposed for object recognition and pose estimation from color images using convolutional neural networks (CNNs). 3D object pose estimation along with object recognition has numerous applications, such as robot positioning vs a target object and robotic object grasping. Previous methods addressing this problem relied on both color and depth (RGB-D) images to learn low-dimensional viewpoint descriptors for object pose retrieval. In the proposed method, a novel quaternion-based multi-objective loss function is used, which combines manifold learning and regression to learn 3D pose descriptors and direct 3D object pose estimation, using only color (RGB) images. The 3D object pose can then be obtained either by using the learned descriptors in a Nearest Neighbor (NN) search, or by direct neural network regression. An extensive experimental evaluation has proven that such descriptors provide greater pose estimation accuracy compared to state-of-the-art methods. In addition, the learned 3D pose descriptors are almost object-independent and, thus, generalizable to unseen objects. Finally, when the object identity is not of interest, the 3D object pose can be regressed directly from the network, by overriding the NN search, thus, significantly reducing the object pose inference time.**

***Index Terms*—3D object pose estimation, convolutional neural networks, multi-objective learning, object recognition, quaternion.**

I. Introduction

OBJECT recognition and 3D pose estimation is a very challenging computer vision task. It has been heavily researched recently, due to its importance in robotics and augmented reality applications. However, there is still a large room for improvement, as occlusion, background clutter, scale and illumination variations highly affect object appearance, and, hence, reduce pose estimation accuracy.

3D object pose estimation typically derives object orientation in a camera coordinate system $(\mathbf{O}_c, X_c, Y_c, Z_c)$, e.g. in a form of a quaternion $\mathbf{q} \in \mathbb{R}^4$. The rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ between the object coordinate system (O_o, X_o, Y_o, Z_o) and the camera coordinate system can be defined by a unit quaternion, as shown in Fig. 1. The 3D object pose estimation problem can be considered as a regression problem, if $\mathbf{q}$ is continuous over $\mathbb{R}^4$ [1]–[3] or as a classification problem, if the 3D pose space has been quantized in a predefined number of orientation classes [4]–[6]. An alternative approach to 3D object pose estimation is transforming the 3D object pose regression problem into a nearest neighbor (NN) one, by matching hand-crafted [7] or extracted [8]–[11] image descriptors with a set of orientation class templates via NN search. It has to be mentioned that the 3D object pose estimation problem addressed by this work is a sub-case of 6D object pose estimation, where both the rotation $\mathbf{R} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{T} \in \mathbb{R}^3$ between the object coordinate system and the camera coordinate system are estimated. Also note that, in this paper, we focus on rigid object pose estimation, and articulated objects are not considered (e.g., human body) [12].

Both classification and regression are typical machine learning problems. Deep learning and especially Convolutional Neural Networks (CNNs) [13] showed remarkable performance in such computer vision tasks, e.g. object detection [14]–[16], recognition [17], [18] and instance segmentation [19]. Deep CNNs were also successfully used for 6D object pose estimation [20]–[24], where both the 3D rotation and 3D translation of the object are estimated. CNNs usually require a huge amount of training data, and there is limited availability of object images annotated with their ground truth 3D pose, due to the inherent difficulty in estimating such a ground truth. However, 3D object models, if available, can be used to create large amounts of synthetic object images along with their ground truth poses for CNN training [8]–[11], [25]. In the proposed method, a lightweight CNN model is trained using both real and synthetic color object images.

Since most pose estimation methods rely on deep network architectures and/or RGB-D data, our goal is to offer a lightweight and reliable RGB only-based 3D object pose estimation method, which can be utilized in embedded systems. Inspired by [10], the proposed method utilizes siamese and triplet CNNs to calculate 3D object pose features. By combining manifold learning and regression, the CNN learns to produce pose features from which both the object identity and 3D pose can be inferred. However, in contrast to [10], the proposed CNN model is forced to learn features, whose distance in the feature space is proportional to the corresponding quaternion distance. To this end, a novel quaternion-based multi-objective loss function is proposed, which combines the strengths of both manifold learning and regression. The trained CNN model demonstrates state-of-the-art 3D object pose estimation accuracy along with object classification. In addition, the object identity and 3D pose are estimated in real time, hence, rendering it suitable for embedded computing in autonomous robotic systems, such as drones. In drone cinematography [26]–[34], 3D target (object) pose estimation is essential for autonomous navigation and visual drone control

Christos Papaioannidis and Ioannis Pitas are with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece. e-mail: cpapaionn@csd.auth.gr, pitas@aiia.csd.auth.gr

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

to achieve the desired cinematography planning objectives, e.g., to get object views from various view angles (poses).

To summarize, this paper offers the following novel contributions:

- introduction of a novel loss function that utilizes quaternions and forces the CNN to learn robust pose features along with direct 3D pose regression;
- improvement in 3D object pose estimation accuracy compared to state-of-the-art (SoA), while using only color images as input (without depth information);
- demonstration of the generalization ability of the proposed method by performing experiments with previously unseen objects;
- examination of various quaternion distance metrics that resemble the real 3D pose differences, to be used in the proposed loss function.

Previous pose estimation methods are reviewed in section II. The proposed QL object pose estimation method and its advantages over previous work are presented in section III. The experimental setup and the extensive evaluation of the proposed method compared to SoA methods can be found in section IV. Finally, conclusions are presented in section V.

II. Related Work

Object recognition and 3D pose estimation have been very active research topics. Early methods were based on sparse feature matching [35] or used 3D point clouds and Point-to-Point matching [36], [37] for 3D object recognition. By using depth information, good 3D object pose estimation results have been achieved for textureless objects [7], [9], [38]–[43]. A template matching framework for object detection and pose estimation from RGB-D images was presented in [7]. However, it was sensitive to occlusions. This approach was later extended to be used in heavily cluttered and occluded scenes by employing Latent-Class Hough Forests [40]. In another direction, random forests were used to estimate the 3D coordinates and the labels of every pixel of the input image, in order to subsequently retrieve the 3D pose [41]. This was extended in [42], by exploiting the uncertainty over pixel 3D coordinates and labels with an auto-context framework to improve 3D pose estimation accuracy.

Recently, CNNs were used to accurately estimate the 3D pose of specific objects. Pre-trained object classification CNNs have been further trained to perform object recognition along with pose estimation from RGB-D images [43]. Pre-trained CNNs were also used in [3] to estimate the camera pose and location from RGB images in an end-to-end manner, while using a quaternion representation for 3D rotations. Also, a pre-trained VGG-M [44] network on ImageNet was used as a base network, coupled with a trained pose estimation network in [45] to perform 3D pose regression. However, a separate pose network for each object of interest is needed in this case. Convolutional auto-encoders were used in [9] to regress descriptors of locally-sampled RGB-D patches for 6D pose estimation.

More recent deep learning methods utilize state-of-the-art object detection [14], [15] and instance segmentation [19] network architectures to the 6D object detection problem. A 6D object pose estimation pipeline was introduced in [21]. Two separate CNNs perform object segmentation and estimation of the 2D location of the 3D object bounding box projections, given the segmentation. The 6D object pose can then be computed using a PnP algorithm [46]. A similar method was used in [23] to regress the 6D object pose using the estimation of the 2D location of the 3D bounding box projections and a PnP algorithm, without the extra CNN segmentation. A different pipeline was used in [22] where a CNN regresses the 2D bounding box of the object. Based on that, it further predicts the object depth, which is used to calculate the object 6D pose. In contrast, the 6D object pose can be directly regressed [20], by extending the Mask-RCNN [19] network architecture by a pose estimation branch. Another interesting approach is introduced in [24], where the SSD [14] object detector is used to detect objects in an image, and then, an Augmented Autoencoder is employed on the detections to learn image features that represent rotations. At test time, the calculated image features are matched with a precomputed codebook using a NN search and the corresponding 3D object poses are returned as estimations.

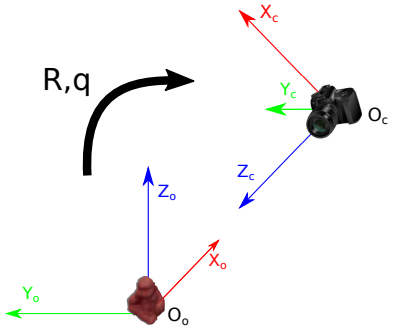
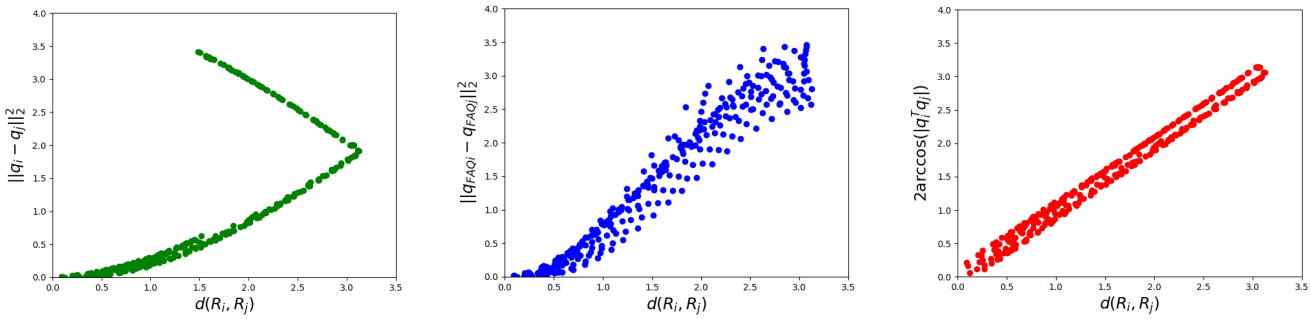


Fig. 1. 3D object pose definition.

A different approach for 3D object pose estimation using lightweight CNNs was introduced in [8], where a framework involving siamese [47], [48] and triplet networks [49] was utilized to learn discriminative features. A NN search is then used on the learned features, in order to obtain the 3D object pose and the object identity label. Later, a dynamic margin was employed in the loss function to improve the robustness of the resulting low-dimensional features [11]. In another approach, the ground truth 3D object poses were used in the optimization process [10] to learn more discriminative features for simultaneous 3D object pose estimation and object recognition. By enforcing a direct relationship between the learned features and the real pose label differences, the model yielded pose features that greatly improved the 3D object pose estimation performance compared to [8]. However, all aforementioned methods rely mostly on RGB-D images to achieve high 3D object pose estimation accuracy. Furthermore, 3D object pose estimation in [8], [10], [11], exclusively employs NN search, which can be slow as the number of objects increases. As depth sensors, in contrast to color cameras, are not always available, e.g. in drones operating outdoors primarily due to cost and weight considerations, a reliable RGB-based 3D object pose



(a) quaternion Euclidean distance (b) 3D cosine distance (c) inverse cosine distance

Fig. 2. The three different quaternion distances plotted against $d(\mathbf{R}_i, \mathbf{R}_j)$.

estimation method is needed.

III. 3D Object Pose Estimation

Let an input image $\mathbf{x} \in \mathbb{R}^D$ (where $D = H \times W \times C$ is the number of elements in a vectorized form, with height H, width W and color channels C) depict an object at a specific pose. The objective of 3D object pose estimation methods is to learn a function $\phi(\mathbf{x}, \mathbf{w})$, in order to map $\mathbf{x}$ to the 3D pose space. The function $\phi(\mathbf{x}, \mathbf{w})$ can be a continuous differentiable CNN function equipped with parameter vector $\mathbf{w}$. To this end, three approaches can be considered. The most obvious one is to train a CNN to regress an object image to its pose, without employing any additional information about the object identity. Alternatively, the same goal can be achieved by classification, if the 3D pose space is quantized to represent 3D pose classes. Finally, another approach is to obtain the pose implicitly, by employing the same CNN architecture as feature extractor, while discarding the regression or classification layer completely. That is, the focus is shifted towards learning lower dimensional 3D pose features, which are then matched with a set of precomputed image database features via NN search [8], [10], [11]. By following this pipeline, besides the 3D object pose, the identity of the object can also be predicted at the same time: the estimated object identity and 3D pose are the ones associated with the retrieved closest image from the database.

A. Unit quaternions and quaternion distance

All possible 3D object orientations in space can be described by the Special Orthogonal Group $\mathcal{SO}(3)$, where each rotation is represented by an orthonormal matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$,

$$\mathbf{R} = \begin{bmatrix} \cos\phi\cos\psi & \cos\phi\sin\psi\sin\theta - \sin\phi\cos\theta & \cos\phi\sin\psi\cos\theta + \sin\phi\sin\theta \\ sin\phi\cos\psi & \sin\phi\sin\psi\sin\theta + \cos\phi\cos\theta & \sin\phi\sin\psi\cos\theta - \cos\phi\sin\theta \\ -\sin\psi & \cos\psi\sin\theta & \cos\psi\cos\theta \end{bmatrix},$$

where ϕ, θ, ψ are the so-called Euler angles that describe 1D rotations around axes X, Y, Z, respectively. Matrix $\mathbf{R}$ satisfies $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det(\mathbf{R}) = 1$. A geometrically meaningful distance metric between two rotation matrices $\mathbf{R}_i, \mathbf{R}_j$ can be defined as follows [50]:

$$d(\mathbf{R}_i, \mathbf{R}_j) = \|\log(\mathbf{R}_i^T\mathbf{R}_j)\|_2, \quad (1)$$

where $\log(\mathbf{R})$ is the matrix logarithm and $\|\cdot\|_2$ is the Frobenius norm. This distance metric measures the length of the shortest path between two points $(\mathbf{R}_i, \mathbf{R}_j)$ on the $\mathcal{SO}(3)$ [50]. However, using rotation matrices to represent rotations in embedded systems has some drawbacks. A rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ consists of 9 variables which are not independent and have only 3 degrees-of-freedom, making the training process difficult. Also, each rotation matrix requires 9 floating point numbers, which can be problematic when limited storage space is available. In addition, estimating rotation matrices can result in matrices that are no longer orthonormal [50], i.e. $\mathbf{R}^T\mathbf{R} \neq \mathbf{I}_3$, due to estimation errors. Training the CNN to estimate the Euler angles (ϕ, θ, ψ) can also be problematic, as the Euler angle representation suffers from the gimbal lock [51] problem, which can cause ambiguity problems during CNN training [2]. Alternatively, every rotation can be determined by its unit rotation axis $\mathbf{u} \in \mathbb{R}^3$, $\mathbf{u} = [u_x, u_y, u_z]^T$ and its rotation angle $\theta \in \mathbb{R}$. Using this so-called axis-angle representation, unit quaternions can be defined. A unit quaternion $\mathbf{q} \in \mathbb{R}^4$, $\mathbf{q} = [q_0, q_1, q_2, q_3]^T = \pm[\cos\frac{\theta}{2}, \mathbf{u}^T\sin\frac{\theta}{2}]^T, \|\mathbf{q}\|_2 = 1$, also represents a rotation of angle θ around the rotation axis $\mathbf{u}$. Unit quaternions offer a preferable alternative for rotations representation, as they offer a more compact representation compared to rotation matrices. Moreover, although estimation errors can also cause a unit quaternion to have magnitude different than 1, it is more straightforward to re-normalize it to unity in comparison to re-normalizing a noisy rotation matrix [50] via SVD. Unit quaternion representation is also preferred over Euler angle representation as it avoids the gimbal lock problem. It has to be noted that unit quaternions double-cover the $\mathcal{SO}(3)$ as $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation. However, by enforcing $q_0 \geq 0$, there is a one-to-one correspondence between rotation matrices and quaternions [52].

In this work, the unit quaternion rotation representation is used, due to its advantages (compact, numerically stable). Therefore, a need for a distance metric that uses unit quaternions is obvious. Three different quaternion distances are examined to find the best approximation of (1), a) the squared Euclidean distance $d_E = \|\mathbf{q}_i - \mathbf{q}_j\|_2^2$, b) the 3D cosine distance, which is simplified to the squared Euclidean distance between *Full-Angle Quaternions* $\mathbf{q}_{faq}$, $d_C = \|\mathbf{q}_{faq_i} - \mathbf{q}_{faq_j}\|_2^2$ [53] and c) the inverse cosine

distance, $d_{IC} = 2\arccos(|\mathbf{q}_i^T \mathbf{q}_j|)$ [11], [50]. Ideally, a linear relationship between the quaternion distance metric and (1) is desirable.

All three quaternion distance metrics are plotted against (1) in Fig. 2. The squared Euclidean distance between unit quaternions cannot be used as a rotation distance metric due to the highly non-linear relation to $d(\mathbf{R}_i, \mathbf{R}_j)$. The 3D cosine distance offer a better rotation distance metric using quaternions but can also be problematic, especially in small pose differences. The inverse cosine distance offers a quaternion distance that best resembles the distance between rotation matrices (1) and thus, is the one used in the proposed method.

B. Previous state-of-the-art pose feature learning methods

The pose feature learning approach offers significant advantages over direct 3D pose regression or classification, as it only requires a lightweight CNN architecture and is scalable to the number of objects [11]. In this case, in order to obtain discriminative and robust pose features, a manifold learning loss function is needed for CNN training. The learned features can then be used in the NN search for 3D object pose and identity retrieval. The overall loss function can be defined as follows:

$$\mathcal{L} = \mathcal{L}_d + \lambda \|\mathbf{w}\|_2^2, \tag{2}$$

where $\mathcal{L}_d$ focuses on feature learning, λ is a regularization parameter and $\|\mathbf{w}\|_2$ is the L_2-norm of the network parameter vector. This framework is based on a training dataset with each sample s_i being of the form $s_i = \{\mathbf{x}_i, c_i, \mathbf{p}_i\}, i = 1, \ldots, N$, where $\mathbf{x}_i$ is the input object image, c_i is the object identity label and $\mathbf{p}_i$ is the 3D object pose. Then, different training samples form sets of pairs $\mathcal{P} = \{s_i, s_j\}$ and triplets $\mathcal{T} = \{s_i, s_j, s_k\}$, which are used in (2) to train the CNN to learn features $\mathbf{f}$, from which the 3D object pose and identity class can be retrieved.

Previous work [8], [10], [11] followed this framework by utilizing siamese and triplet networks to learn features for 3D object pose estimation. Manifold learning for 3D pose estimation was first introduced in [8], by employing the overall loss function:

$$\mathcal{L} = \mathcal{L}_{pairs} + \mathcal{L}_{triplets} + \lambda \|\mathbf{w}\|_2^2. \tag{3}$$

If sample $s_i = \{\mathbf{x}_i, c_i, \mathbf{p}_i\}$ is an anchor sample coming from the training dataset, sample pairs $\{s_i, s_j\}$ coming from the same object and corresponding to very similar poses contribute to the pairwise loss:

$$\mathcal{L}_{pairs} = \sum_{(s_i, s_j) \in \mathcal{P}} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2. \tag{4}$$

Also, triplets $\{s_i, s_j, s_k\}$ were defined in the following way:

- either s_i, s_j belong to the same object and s_k to any different object or,
- all three samples s_i, s_j, s_k belong to the same object, with $\mathbf{p}_i$, $\mathbf{p}_j$ being similar 3D poses, while $\mathbf{p}_i$, $\mathbf{p}_k$ being dissimilar ones.

Triplets were used to define the triplet loss $\mathcal{L}_{triplets}$ as follows [8]:

$$\mathcal{L}_{triplets} = \sum_{(s_i, s_j, s_k) \in \mathcal{T}} max\left(0, 1 - \frac{\Delta_-}{\Delta_+ + \varepsilon}\right), \tag{5}$$

where ε is a margin value and $\Delta_+ = \|\mathbf{f}_i - \mathbf{f}_j\|_2$, $\Delta_- = \|\mathbf{f}_i - \mathbf{f}_k\|_2$ are the Euclidean feature distances between the similar (s_i, s_j) and dissimilar (s_i, s_k) samples in the triplet $\{s_i, s_j, s_k\}$.

As indicated by (3), a direct pose regression term is absent. Therefore, the estimated pose can only be obtained via NN search. Furthermore, the absence of a regression loss term in the total loss function leads to inferior performance, as demonstrated in [10].

The total loss function (3) introduced by [8] was extended in [11], where a dynamic margin ε_d was used in (5) instead of the static ε. The dynamic margin ε_d, which utilized unit quaternions and the inverse cosine distance, was defined as:

$$\varepsilon_d = \begin{cases} 2\arccos(|\mathbf{q}_i^T \mathbf{q}_j|) & if\ c_i = c_j, \\ n & \text{else, for } n > \pi. \end{cases} \tag{6}$$

It should be noted that in both works [8], [11] the real sample 3D poses were not utilized in the training process, in contrast to [10], where the total loss function is given by:

$$\mathcal{L} = \mathcal{L}_{pose} + \mathcal{L}_{object} + \mathcal{L}_{regression} + \lambda \|\mathbf{w}\|_2^2, \tag{7}$$

having the regression term $\mathcal{L}_{regression}$. The pairwise loss $\mathcal{L}_{pose}$ used sample pairs $\{s_i, s_j\}$ from the same object that may have different 3D poses:

$$\mathcal{L}_{pose} = \sum_{s_i, s_j} \{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2 - \delta(\mathbf{p}_i, \mathbf{p}_j)\}^2, \tag{8}$$

where δ is the squared Euclidean distance $\|\mathbf{p}_i - \mathbf{p}_j\|_2^2$ between the ground truth poses $\mathbf{p}_i, \mathbf{p}_j$. In addition, depth information was also used to weigh the contribution of each sample pair. $\mathcal{L}_{pose}$ was used to impose 3D object pose difference information on the pose feature vectors $\mathbf{f}_i$, $\mathbf{f}_j$. $\mathcal{L}_{object}$ used triplets $\{s_i, s_j, s_k\}$ in order to train the model to deal with samples coming from different objects and was defined as:

$$\mathcal{L}_{object} = \sum_{s_i, s_j, s_k} \frac{\Delta_+}{\Delta_- + \varepsilon}, \tag{9}$$

where s_i, s_j belong to the same object identity and s_k to any other object identity. The regression loss, $\mathcal{L}_{regression}$, used in their method was a RMS error $\|\mathbf{p} - \hat{\mathbf{p}}\|_2^2$ over all samples, again weighted by a depth-related term, where $\mathbf{p}$, $\hat{\mathbf{p}}$ is the ground truth and the regressed 3D object pose, respectively.

C. Proposed QL object pose estimation method

The total loss function used in the proposed quaternion learning (QL) 3D object pose estimation method is:

$$\mathcal{L} = \mathcal{L}_{desc} + \mathcal{L}_{qreg} + \lambda \|\mathbf{w}\|_2^2. \tag{10}$$

The novel loss function $\mathcal{L}_{desc}$ aims at learning robust features from which, both the object identity and the 3D pose can be inferred. The quaternion regression loss function $\mathcal{L}_{qreg}$, apart

from allowing the network to directly predict object 3D poses, also enhances feature learning, by inferring extra information about the 3D object poses in the optimization process.

Unit quaternion regression needs some special care, as the four quaternion entries q_0, q_1, q_2, q_3 are not independent. The term $\sin\frac{\theta}{2}$ is found in all three entries q_1, q_2, q_3, while $\cos\frac{\theta}{2}$ contributes to q_0. As a result, direct regression to unit quaternions leads to inferior performance [2], as it is difficult for the CNN to learn this dependence between q_0, q_1, q_2, q_3. In contrast, in the proposed method, the independent axis-angle rotation representation entries $\mathbf{r} = [\theta', u_1, u_2, u_3]^T$, $\theta' = \frac{\theta}{2}$ are regressed, as depicted in Fig. 3. Ultimately, the 3D pose quaternion regression error can be defined as follows:

$$\mathcal{L}_{qreg} = \|\mathbf{q} - \hat{\mathbf{q}}\|_2^2, \tag{11}$$

where:

$$\begin{cases} \hat{q}_0 = \cos(\theta') \\ \hat{q}_1 = u_1 \sin(\theta') \\ \hat{q}_2 = u_2 \sin(\theta') \\ \hat{q}_3 = u_3 \sin(\theta'). \end{cases} \tag{12}$$

It is worth mentioning that the output of the network $\hat{\mathbf{q}}$ is not strictly forced to have magnitude 1 (e.g. by applying L_2 normalization), as it makes the CNN training harder [2]. Nevertheless, by using unit quaternions as labels, the regressed $\hat{\mathbf{q}}$ norm does not diverge too much from the unit norm. During testing, L_2 normalization can be applied to ensure unit quaternion estimation.

Note that, the inverse cosine distance $2arccos(|\mathbf{q}^T\hat{\mathbf{q}}|)$ could have been used as the regression loss function $\mathcal{L}_{qreg}$, since it is proven to be a better quaternion distance metric (III-A). However, in the case of using the inverse cosine distance for regression, the partial derivative of $2arccos(|\mathbf{q}^T\hat{\mathbf{q}}|)$ with respect to $\hat{q}_i$ needs to be calculated. This derivative causes problems during training. More specifically, the gradient is not continuous in the interval $(-1, 1)$ at point 0 and gets extreme values at the points where $2arccos(|\mathbf{q}^T\hat{\mathbf{q}}|) \to 0$, i.e., its optimal position, as can be seen below:

$$\frac{\partial}{\partial \hat{q}_i}(2arccos(|\mathbf{q}^T\hat{\mathbf{q}}|)) = -\frac{2q_i(\mathbf{q}^T\hat{\mathbf{q}})}{\sqrt{1-(\mathbf{q}^T\hat{\mathbf{q}})^2}|(\mathbf{q}^T\hat{\mathbf{q}})|}, \tag{13}$$

where q_i is an element of $\mathbf{q}$. The proposed regression loss function (11) in fact minimizes the inverse cosine distance indirectly. $\mathcal{L}_{qreg}$ is minimized when $\hat{\mathbf{q}}$ approaches $\mathbf{q}$, and since $\mathbf{q}$ is a unit quaternion, the dot product between $\mathbf{q}$ and $\hat{\mathbf{q}}$ will be close to 1. Therefore, minimizing (11) leads to minimizing the inverse cosine distance as well, avoiding convergence issues.

3D pose descriptor learning requires a discriminative feature space, in order to exploit the learned features in the NN search and obtain accurate 3D object pose and identity estimation. Ideally, features coming from different 3D objects should be distinct, forming distinguishable and compact classes. Moreover, the distance between features of samples with similar poses should be small and their distance for ones with dissimilar poses should be large, in order to form 3D object pose sub-clusters within the object identity classes. To accomplish these requirements, a triplet loss function must be utilized in conjunction with a pairwise loss function for

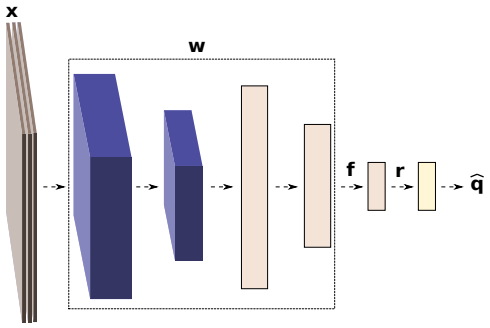


Fig. 3. CNN architecture for 3D object pose regression.

CNN training. The triplet loss function ensures the formation of object clusters in the feature space, while the pairwise loss function is used to infer 3D pose similarity or dissimilarity.

In order for such loss functions to be used, the data used for CNN training $\mathcal{S}_{train} = \{s_1, \ldots, s_N\}$ consist of N data samples, where each sample $s_i = \{\mathbf{x}_i, c_i, \mathbf{q}_i\}, i = 1, \ldots, N$ contains an RGB image $\mathbf{x}_i$ of an object, its assigned object identity label $c_i \in \mathcal{C} = \{c_1, \ldots, c_L\}$ and the corresponding 3D pose quaternion $\mathbf{q}_i \in \mathbb{R}^4$. Let $\mathcal{P} = \{s_i, s_j\}$, $\mathcal{T} = \{s_i, s_j, s_k\}$ be sets containing training sample pairs and triplets, respectively. The proposed feature learning loss function is of the following form:

$$\mathcal{L}_{desc} = \mathcal{L}_p + \mathcal{L}_o \tag{14}$$

The pairwise loss $\mathcal{L}_p$ is computed on pairs $\{s_i, s_j\} \in \mathcal{P}$, where the samples s_i, s_j belong in the same object identity $c_l, l = 1, \ldots, L$. $\mathcal{L}_p$ is used to enforce pose similarity within the same object identity c_l. Therefore, if $\mathbf{f}_i = f(\mathbf{x}_i) \in \mathcal{F} \subset \mathbb{R}^d$ are the features obtained from the last fully connected CNN layer having $\mathbf{x}_i$ as input, the pairwise loss is defined as:

$$\mathcal{L}_p = \sum_{s_i, s_j} \{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2 - 2\arccos(|\mathbf{q}_i^T\mathbf{q}_j|)\}^2, \tag{15}$$

where $|\mathbf{q}_i^T\mathbf{q}_j|$ is the absolute value of the inner product between the ground truth 3D object pose quaternions $\mathbf{q}_i$, $\mathbf{q}_j$. Essentially, $\mathcal{L}_p$ forces the Euclidean feature distance between two samples from the same object identity to be equal to the quaternion distance between the corresponding 3D poses $\mathbf{q}_i$, $\mathbf{q}_j$. Thus, if the quaternion distance is small, the optimization process forces the corresponding feature distance to be small as well. The same applies if the quaternion distance between the pair 3D poses is large. Therefore, $\mathcal{L}_p$ leads to the formation of pose-related clusters in the feature space $\mathcal{F}$. Based on this, the features obtained from the proposed method will be referred as 3D pose descriptors and, thus, the feature space $\mathcal{F}$ as descriptor space.

The triplet loss term $\mathcal{L}_o$ in (14) enforces the 3D pose descriptors coming from same-object class samples to have smaller distances in the descriptor space, when compared to the distances of descriptors calculated from different object identity samples. For this purpose, the sample triplets $\{s_i, s_j, s_k\} \in \mathcal{T}$, consist of samples s_i, s_j coming from the same object identity $c_l, l = 1, \ldots, L$, while s_k is a sample coming from any different object identity. $\mathcal{L}_o$ is similar to (9) so that the distance in the descriptor space between the

TABLE I
TOTAL LOSS FUNCTIONS USED IN THE METHODS OF [8], [10], [11] AND THE PROPOSED METHOD.

[8]	$\mathcal{L} = \sum \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 + \sum max\big(0, 1 - \frac{\Delta_-}{\Delta_+ + \varepsilon}\big) + \lambda\|\mathbf{w}\|_2^2$
[11]	$\mathcal{L} = \sum \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 + \sum max\big(0, 1 - \frac{\Delta_-}{\Delta_+ + \varepsilon_d}\big) + \lambda\|\mathbf{w}\|_2^2$, $\varepsilon_d = \begin{cases} 2\arccos(\lvert\mathbf{q}_i^T\mathbf{q}_j\rvert) & if\ c_i{=}c_j, \\ n & else, for\ n{>}\pi \end{cases}$
[10]	$\mathcal{L} = \sum\{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2 - \|\mathbf{p}_i - \mathbf{p}_j\|_2^2\}^2 + \sum \frac{\Delta_+}{\Delta_- + \varepsilon} + \sum \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2^2 + \lambda\|\mathbf{w}\|_2^2$
ours	$\mathcal{L} = \sum\{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2 - 2\arccos(\lvert\mathbf{q}_i^T\mathbf{q}_j\rvert)\}^2 + \sum \frac{\Delta_+}{\Delta_- + \varepsilon} + \sum \|\mathbf{q}_i - \hat{\mathbf{q}}_i\|_2^2 + \lambda\|\mathbf{w}\|_2^2$

same object identity is forced to be smaller than the distance between object descriptors coming from different classes. ε is a small regularizing constant, that also prevents having a zero denominator in (9). Essentially, $\mathcal{L}_o$ minimizes the within object class cluster distance in the descriptor space.

The critical difference of the proposed QL object pose estimation method compared to previous work is that we use the quaternion distance between the ground truth 3D object poses in the pairwise loss function (15) instead of the Euclidean distance δ used in [10]. Furthermore, inspired by [2], a regression loss function (11) that enables 3D pose quaternion estimations is also used, in conjunction with (14) for CNN training. These two novelties provide superior 3D object pose regression performance, as shown in the next sections. The total loss functions used in [8], [10], [11] and the proposed method are presented in Table I.

IV. EXPERIMENTAL RESULTS

In this section, a detailed description of the dataset generation is given, along with implementation details. In addition, the employed baseline models and performance evaluation, are presented.

A. Object pose dataset generation

Since all related methods [8], [10], [11] used the LineMOD dataset [54] for their experiments, all our experiments were performed using the same dataset, to ensure a fair comparison.

The LineMOD dataset consists of RGB-D sequences of fifteen different every-day objects, along with their 3D poses. In addition, for each object, a 3D mesh model is available. These data were used to create three separate sets, the training $\mathcal{S}_{train}$, template $\mathcal{S}_{templ}$ and test $\mathcal{S}_{test}$ sets, respectively. In the proposed method, each set consists of samples $s = \{\mathbf{x}, c, \mathbf{q}\}$ where $\mathbf{x}$ is only the RGB object image (by dropping depth information) and c, $\mathbf{q}$ are the corresponding object identity label and the assigned ground truth 3D pose quaternion, respectively. The $\mathcal{S}_{train}$ set is used in the training process of the network. The template $\mathcal{S}_{templ}$ set is used in 3D object pose estimation performance evaluation, where it acts as a database: its elements are matched to the test images via NN search. The $\mathcal{S}_{test}$ set is used only in the test stage, where the trained network is evaluated using the appropriate performance evaluation metrics. It has to be noted that $\mathcal{S}_{train}$ set contains a mixture of real and synthetic RGB images, while $\mathcal{S}_{test}$, $\mathcal{S}_{templ}$ consist only of real and synthetic images, respectively. Synthetic images were rendered from the corresponding 3D mesh models, as described below.

The synthetic data were created by rendering the available object mesh models by positioning a virtual camera at various viewpoints on a half dome over the 3D object model [8]. At first, the camera viewpoints were at the vertices of a regular icosahedron. By recursively subdividing each triangle into 4 sub-triangles, more viewpoints were defined, and thus, a denser viewpoint sampling was created. For the $\mathcal{S}_{templ}$ database viewpoints, the subdivision was applied only two times, resulting in 301 evenly distributed viewpoints. For the synthetic data used in the training set, the subdivision was performed one more time, ending up in 1241 different viewpoints. The object was then rendered as seen from each such viewpoint. After rendering the synthetic images of $\mathcal{S}_{train}$, background fractal noise [55] was added. Using fractal noise to simulate backgrounds is a common technique [8], [10], [11] and it was proven to be the most suitable type of synthetic noise to be used as background, when the test set environment is not known beforehand [11]. Since it is desirable our model to generalize well to different image domains and fractal noise was also used as background for synthetic images in the most similar methods [8], [10], we also use fractal noise as background for the synthetic images.

The real world data were included both in the training and test datasets, by ensuring a uniform viewpoint distribution over the viewing domain (hemisphere). The samples were roughly 50%-50% split over the training and test set, as in [8]. Given that the camera intrinsic parameters were provided with the dataset, RGB image patches $\mathbf{x}$ were extracted from a bounding cube centered at the object center and all values beyond this bounding cube were clipped. These image patches were then stored along with their objects class c and the corresponding 3D pose quaternion $\mathbf{q}$, forming a sample s.

It has to be noted that the synthetic images used in $\mathcal{S}_{templ}$ have no added noise, as the network should map the noisy synthetic and the real world input images to the same location in the descriptor space, with the map of clean template images. Examples of real and synthetic images are shown in Fig. 4.

Rotationally invariant objects. Some of the LineMOD dataset objects are rotationally invariant to different degrees, as also pointed out in previous work [8], [10]. Specifically, the objects *bowl*, *cup*, *eggbox* and *glue* needed to be treated differently from the rest, when creating our pairs and triplets for the training process. Similar to [8], we treated the *bowl* object as fully rotationally invariant, meaning that the azimuth of the viewing angle should not be considered, as the bowl appearance is the same over all different azimuth angles. Moreover, the objects *eggbox* and *glue* were treated

TABLE II
3D OBJECT POSE ESTIMATION AND OBJECT CLASSIFICATION ACCURACY.

	Angular threshold t								
	5°	**10°**	**15°**	**20°**	**30°**	**40°**	**45°**	**Mean (Median) ± Std**	**Object classification**
$3DPOD$ [8]	36.47%	64.11%	77.32%	83.87%	89.37%	91.78%	92.71%	16.38°(8.10°) ± 27.24°	96.11%
$PEDM$ [11] ⋆	-	60.00%	-	93.20%	-	98.00%	-	-	99.30%
$PGFL$ [10]	37.19%	**80.30%**	92.73%	96.26%	98.49%	99.14%	99.26%	7.61°(6.12°) ± 8.57°	98.80%
QL (ours)	**40.15%**	79.42%	**93.66%**	**97.77%**	**99.63%**	**99.93%**	**99.95%**	**6.87°(5.91°) ± 5.08°**	98.80%
$PGFL_d$ [10] †	34.11%	**75.02%**	90.53%	95.36%	98.35%	98.96%	99.20%	8.07°(6.51°) ± 8.78°	97.23%
QL_d (ours) †	**36.13%**	74.05%	**91.15%**	**96.72%**	**99.34%**	**99.74%**	**99.79%**	**7.51°(6.38°) ± 6.32°**	97.78%

⋆ The results of $PEDM$ are directly cited from [11] with RGB-D setting.
† Refers to training the models without regression loss.

(a) real (b) synthetic (c) template (d) cyclist

Fig. 4. Examples of images used for CNN training and testing.

as 180° symmetric around the z-axis. As also stated in [8], the *cup* object is a special case, because, when the handle is not visible, it can be treated as a rotationally invariant one. However, this is not always the case, as, for example, when its handle is visible. In this work, we also treated it as rotationally invariant, as in [8].

B. CNN implementation details

The CNN used in the proposed method has the same architecture to [8], [10], [11], while, in our case, the input layer has only 3 channels (RGB). The first two network layers are convolutional ones with max-pooling and a rectified linear (ReLU) activation function, followed by two fully connected layers. The final fully connected layer produces the 3D pose descriptor vector $\mathbf{f} \in \mathcal{F} \subset \mathbb{R}^d$. In all the experiments, the descriptor dimensionality was set to $d = 32$, as we have seen no benefit by its further increase. For the quaternion regression, an extra fully connected layer is added after the descriptor layer. Then, as depicted in Fig. 3, this layer is followed by the quaternion activation layer, which maps the regressed $\mathbf{r}$ vector to $\hat{\mathbf{q}}$, according to (12). The overall CNN is trained using the stochastic gradient decent method [56], with momentum 0.9 and initial learning rate of 0.01, as in [10]. The learning rate is reduced in every epoch and the batch size was set to 120. Keras [57] with Tensorflow [58] backend, were used for the proposed method implementation.

C. Evaluation procedure

In all experiments the models were trained using the LineMOD dataset as modified by [8]. The proposed method is compared to the baseline methods of [8], [10], [11]. It has to be noted, that the method of [10] was implemented in Python, as the code was not publicly available. At first, a CNN model was trained by following the method described in [10] and using the RGB-D sequences of the LineMOD dataset. This was only to ensure that our implementation of [10] had a similar performance with the one reported in [10], which is based on RGB-D data. Then, by using the same optimization framework, a different model was trained from scratch, this time using only RGB images as inputs and by omitting any depth-related information in the total loss function. This RGB-based CNN model is denoted by $PGFL$, where the estimated 3D object pose is obtained by NN search. In addition to the $PGFL$ model, which was trained using the total loss function (7), another model was trained using again RGB images as inputs and only the $\mathcal{L}_{pose}$, $\mathcal{L}_{object}$ terms. This new CNN model is referred as $PGFL_d$. Similarly, two different CNN models of the proposed method were trained, denoted as QL and QL_d. QL is trained with the full multi-objective loss function (10) and the QL_d model is trained using only the proposed pose descriptor learning term $\mathcal{L}_{desc}$ in (10) (without the quaternion regression term $\mathcal{L}_{qreg}$). Moreover, using the available code for [8], a model was trained with the RGB setting and is denoted by $3DPOD$. Also, the method of [11] is denoted by $PEDM$. Note that, in all models reported in Table II, the estimated 3D object pose is the ground truth pose assigned to the closest $\mathcal{S}_{templ}$ sample retrieved by the nearest neighbor search.

Given a test sample $s = \{\mathbf{x}, c, \mathbf{q}\}$ coming from $\mathcal{S}_{test}$, the 3D object pose estimation error between the ground truth pose $\mathbf{q}$ and the corresponding estimated pose $\hat{\mathbf{q}}$, is the angular error in degrees, calculated using the inverse cosine distance:

$$err(\mathbf{q}, \hat{\mathbf{q}}) = 2 \arccos(|\mathbf{q}^T \hat{\mathbf{q}}|). \tag{16}$$

The 3D pose estimation accuracy at threshold t is then defined as the percentage of test samples, for which the angular error between the estimated and the ground truth pose is below a threshold angle t, $err(\mathbf{q}, \hat{\mathbf{q}}) < t$. It should be noted that, the pose estimation accuracy is calculated only for the test samples that were correctly matched to their corresponding object identity. A comparison of the performance between the proposed CNN model QL and the baseline CNN models $3DPOD$, $PEDM$ and $PGFL$ for threshold angle values

TABLE III
3D OBJECT POSE ESTIMATION ACCURACY OF THE PROPOSED QL AND $PGFL$ [10] METHODS FOR EACH LINEMOD OBJECT, FOR ANGLE THRESHOLDS $t \in [5^o, 15^o, 30^o]$.

		ape	benchv	bowl	cam	can	cat	cup	driller	duck	eggbox	glue	holep	iron	lamp	phone
5°	$PGFL$ [10]	**33.53%**	**34.57%**	72.74%	**36.32%**	**35.42%**	32.35%	33.14%	**29.87%**	**37.68%**	37.13%	34.41%	29.74%	**35.71%**	**36.82%**	**37.82%**
	QL (ours)	32.56%	32.23%	**97.70%**	33.43%	32.40%	**35.80%**	**39.91%**	27.23%	31.06%	**49.02%**	**48.66%**	**31.71%**	34.58%	34.00%	37.72%
15°	$PGFL$ [10]	**95.05%**	91.90%	98.66%	**95.36%**	91.22%	91.48%	93.63%	**86.63%**	91.06%	89.84%	96.29%	91.35%	**92.20%**	91.46%	**94.63%**
	QL (ours)	93.83%	**93.10%**	**100%**	93.95%	**91.24%**	**91.86%**	**95.50%**	85.92%	**91.44%**	**98.04%**	**99.10%**	**92.86%**	91.88%	**92.53%**	92.63%
30°	$PGFL$ [10]	99.40%	98.75%	99.85%	98.80%	98.11%	98.45%	99.42%	94.96%	99.12%	97.91%	99.53%	98.58%	98.70%	96.10%	99.42%
	QL (ours)	**99.69%**	**100%**	**100%**	**100%**	**99.53%**	**99.85%**	**100%**	**97.65%**	**99.71%**	**100%**	**100%**	**99.43%**	**99.03%**	**99.54%**	**99.86%**

$t \in [5°, 10°, 15°, 20°, 30°, 40°, 45°]$ is shown in Table II, where the object classification accuracy is also reported. It should be noted that, since the code of [11] could not be made available, the results reported in [11] are directly cited in Table II only for threshold angle values $t \in [10°, 20°, 40°]$. The proposed method improves the 3D object pose estimation accuracy, particularly when high pose estimation accuracy is needed, e.g. $err(\mathbf{q}, \hat{\mathbf{q}}) < 5°$. Also, note that, if we are just interested on coarse 3D object pose estimation, i.e., by classifying it to *frontal*, *side*, *back*, *top* views, the threshold $t = 45°$ should be used.

To further evaluate the 3D object pose estimation performance of the proposed method, the mean and standard deviation values of the pose estimation error (16), are also presented in Table II. These values were also calculated using the estimations of the proposed and all baseline CNN models for samples coming from $\mathcal{S}_{test}$, that were correctly classified to their object identity. The proposed model QL have lower mean and standard deviation values of the angular error compared to all baseline models.

The effect of using the quaternion distance in (15) can be seen by comparing the performance of $PGFL_d$ and QL_d reported in Table II. The quaternion distance used in the pairwise loss, boosted the 3D object pose estimation performance both in terms of accuracy and mean angular error. Since the quaternion distance used in the proposed method is a better metric for 3D pose differences, the similar-pose clusters formed in the descriptor space by $\mathcal{L}_p$ are more compact, allowing the network to more successfully retrieve the closest $\mathcal{S}_{templ}$ sample. Also, the object classification rate is slightly increased, meaning that the object identity clusters in the descriptor space were less affected by the quaternion distance. Moreover, the comparison between QL_d and QL shows that, when the quaternion regression term $\mathcal{L}_{qreg}$ is also included in the total loss function during the training process (QL), the 3D object pose estimation accuracy, the mean angular error and the object classification results are further improved. As the pose estimations are obtained by NN search on the descriptor space, this means that $\mathcal{L}_{qreg}$, when combined with $\mathcal{L}_{desc}$, forces the CNN to learn more robust 3D pose descriptors.

The 3D pose estimation accuracy for each object in the LineMOD dataset is presented in Table III. The comparison is conducted between the proposed method QL and the second best performing method $PGFL$ for threshold values $t \in [5^o, 15^o, 30^o]$, which cover the high, medium and low accuracy areas, respectively. The reported results show that the performance for the rotationally invariant objects (*bowl*, *cup*, *glue*, *eggbox*) is increased, especially in the high accuracy threshold ($t = 5^o$). This can be explained by the fact that the model should only learn to encode the elevation in the case of fully rotationally invariant objects (*bowl*), or half of the azimuth angles and the elevation of the objects that are 180^o symmetric around the z-axis (*cup*, *glue*, *eggbox*). These results also highlight the need to treat non-trivial object symmetries by using only RGB images. In addition, the comparison between the performance of the proposed method QL and $PGFL$ for each object, show that the proposed loss function better imposes the 3D pose information to the model, as the accuracy of QL at threshold $t = 30^o$ is nearly 100%, for all objects. Moreover, the proposed method accuracy is over 95% for all rotationally invariant objects even in the medium accuracy threshold ($t = 15^o$).

The quaternion regression term (11), not only enhances the 3D object pose estimation performance, but also offers the CNN the ability to directly regress the 3D object pose. By directly regressing the 3D object pose, the NN search is completely omitted and, thus, faster 3D object pose estimations

TABLE IV
PROPOSED METHOD 3D OBJECT POSE ESTIMATION ACCURACY BY CNN REGRESSION OR VIA NN SEARCH.

	Angular threshold t					
	5°	**15°**	**30°**	**45°**	**Mean (Median) ± Std**	**Inference time**
QL_R	21.18%	72.53%	86.79%	89.93%	23.32°(8.99°) ± 42.07°	**2.8** *ms*
QL	**40.15%**	**93.66%**	**99.63%**	**99.95%**	**6.87°(5.91°) ± 5.08°**	5.3 *ms*

TABLE V
3D OBJECT POSE ESTIMATION PERFORMANCE ON AN UNSEEN OBJECT.

	Angular threshold t				
	5°	**15°**	**30°**	**45°**	**Mean (Median) ± Std**
$PGFL$ [10]	31.99%	59.31%	73.57%	77.68%	33.28°(10.57°) ± 48.70°
QL (ours)	**49.00%**	**82.35%**	**90.81%**	**93.23%**	**15.88°(5.06°) ± 35.37°**
$PGFL_d$ [10] †	30.37%	53.26%	64.94%	71.79%	40.10°(12.59°) ± 53.15°
QL_d (ours) †	**36.26%**	**63.17%**	**76.14%**	**83.00%**	**29.16°(8.58°) ± 46.79°**

† Refers to training the models without regression loss.

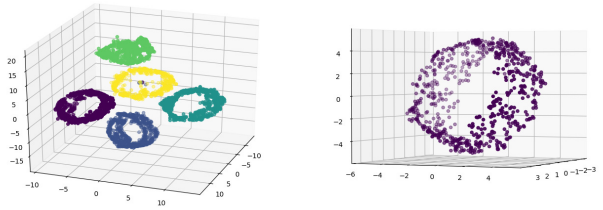
are possible. When the estimated 3D object pose is obtained by direct regression, the CNN model is denoted as QL_R. Note that, QL_R and QL are the same CNN model, with the only difference being that during testing, the 3D object pose estimations are obtained by direct network regression in the case of QL_R, instead of employing a NN search on the learned pose descriptors (QL). As shown in Table IV, the 3D object pose estimation performance of QL_R is significantly lower compared to the performance of QL that uses NN search. Nevertheless, direct 3D object pose regression can be effectively used in cases where only coarse 3D object pose estimation is desirable (e.g. for $t = 45°$). Also, it must be noted that QL_R speed is nearly double that of QL, as shown in Table IV. Computational speed was calculated using *Ubuntu* and a *GeForce GTX 1080 Ti* graphics card.

D. Generalization ability to unseen objects

In addition to the experiments with the LineMOD dataset, an evaluation of the generalization ability of the proposed QL method to previously unseen objects was performed. To this end, the best performing CNN models QL, $PGFL$ and QL_d, $PGFL_d$, which were trained using the full LineMOD dataset, were tested on synthetic *cyclist* images, like the one shown in Fig. 4d. These images were rendered using a 3D mesh model and the pipeline described in IV-A. The choice of synthetic *cyclist* images as test images was made in order to evaluate the ability of the CNN models to generalize to unseen objects having more complex appearance (cyclist on a bicycle) than the ones used in training.

As shown in Table V, both CNN models QL, QL_d, obtained from the proposed method, greatly outperform the baseline ones $PGFL$, $PGFL_d$, proving again that the pose descriptors learned by the proposed quaternion-based loss function better encapsulate the 3D object pose information. In particular, when comparing the performance between QL and $PGFL$, a great increase of the pose accuracy is observed for all threshold values, especially in the high accuracy thresholds $t = 5°$ and $t = 15°$, where the 3D object pose estimation accuracy increase is 17% and 23%, respectively. As also shown in Table V, the mean and standard deviation angular error values calculated for the proposed QL method pose estimations are significantly smaller than the ones calculated for the baseline method. It should be noted that the object classification rate in these experiments is 100%, as only *cyclist* object samples were included in the $\mathcal{S}_{templ}$ set. This was to specifically evaluate the 3D object pose estimation performance of both methods on an unseen object.

The results obtained by the CNN models QL_d and $PGFL_d$ shown in Table V, show the impact of the regression terms $\mathcal{L}_{qreg}$, $\mathcal{L}_{regression}$ on performance, when it comes to unseen objects. When the the total loss function (10) is used for CNN training (QL), the 3D object pose estimation accuracy is improved by 12.74% for the angle accuracy threshold $t = 5°$ and by 19.18% for $t = 15°$ compared to QL_d performance. The corresponding increase obtained by using the total loss function (7) of the baseline method $PGFL$, when compared to $PGFL_d$ model, is only 1.62% and 6.05% for $t = 5°$ and $t = 15°$, respectively. These results prove that the proposed quaternion regression loss function $\mathcal{L}_{qreg}$ makes the CNN model more sensitive to 3D object pose differences, regardless the object.



(a) five objects (b) single object

Fig. 5. Descriptor visualization of test images in 3D space.

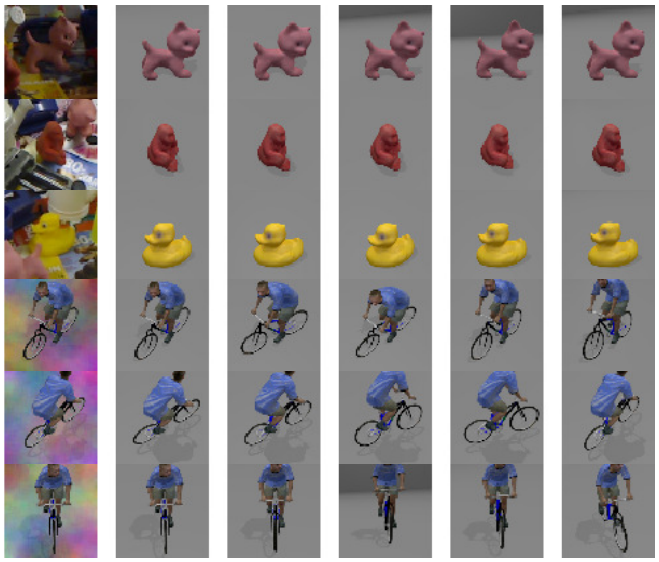


Fig. 6. Retrieved top 5 nearest neighbors for 6 query (test) images. Left column shows the query images and the rest columns depict the retrieved closest nearest neighbors from left to right.

E. Qualitative evaluation

Apart from the 3D object pose estimation accuracy reported in Tables II - V, a qualitative evaluation of the proposed method is also performed. This is to give a more intuitive demonstration of the 3D object pose estimation capabilities of the proposed QL object pose estimation method.

At first, we present a visualization of the learned 3D pose descriptors $\mathbf{f}_i$. Such descriptors of test images coming from five of the fifteen different objects in the LineMOD dataset are depicted in Fig. 5a, in a 3D dimensional space. The selection of five random objects was made only for visualization purposes. The dimensionality reduction from $d = 32$ to 3, was performed using the t-SNE algorithm [59] provided by the Python library *scikit-learn* [60]. All five object identity clusters appear to be easily distinguishable. A more clear view of each of the formed class clusters is given in Fig. 5b, where the learned 3D pose descriptors of test images coming from a single random object are shown. Each point represents an image depicting the object at a specific pose.

In addition, the images of the top 5 $\mathcal{S}_{templ}$ samples retrieved by NN search for each query image, are presented in Fig. 6. The query images either belong to the test set $\mathcal{S}_{test}$ or are synthetic images of the unseen *cyclist* object. In both cases, the proposed method matches the query images to $\mathcal{S}_{templ}$ sample images that have very similar 3D pose, with the 3D pose difference between them being imperceptible in most cases.

V. Conclusion

In this work, a framework for 3D object pose estimation along with object recognition using a lightweight CNN was presented. In contrast to previous work, it is proven that RGB images are sufficient for accurate 3D object pose estimation without using depth information. This fact allows various applications in robotics, e.g., view selection for drone cinematography. By examining the most appropriate 3D pose distance metric, a multi-objective loss function based on quaternions, is proposed. The proposed QL object pose estimation method yielded more discriminating 3D pose descriptors, hence increasing the 3D object pose accuracy compared to the state-of-the-art. It also provided the CNN generalization ability to unseen objects. In addition, when the object identity is not important, the 3D object pose can be directly regressed from the CNN, thus, reducing the 3D object pose inference time.

Future work can extend this method to take into account object symmetries that are non-trivial, while using only RGB images. In addition, the ability of the CNN model to generalize to different image domains should also be examined, as it would allow training models only with synthetic images. Finally, training the CNN model to treat heavily occluded images should also be considered.

References

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

[2] H.-W. Hsu, T.-Y. Wu, S. Wan, W. H. Wong, and C.-Y. Lee, "Quatnet: Quaternion-based head pose estimation with multi-regression loss," *IEEE Transactions on Multimedia*, 2018.

[3] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2938–2946.

[4] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2686–2694.

[5] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1510–1519.

[6] T. Chen and S. Lu, "Robust vehicle detection and viewpoint estimation with soft discriminative mixture model," *IEEE Transactions on circuits and systems for Video technology*, vol. 27, no. 2, pp. 394–403, 2017.

[7] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Asian conference on computer vision*. Springer, 2012, pp. 548–562.

[8] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3109–3118.

[9] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 205–220.

[10] V. Balntas, A. Doumanoglou, C. Sahin, J. Sock, R. Kouskouridas, and T.-K. Kim, "Pose guided rgbd feature learning for 3d object pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3856–3864.

[11] S. Zakharov, W. Kehl, B. Planche, A. Hutter, and S. Ilic, "3d object instance recognition and pose estimation using triplet loss with dynamic margin," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 552–559.

[12] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1653–1660.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[20] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6dpose: Recovering 6d object pose from a single rgb image," *arXiv preprint arXiv:1802.10367*, 2018.

[21] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *International Conference on Computer Vision*, vol. 1, no. 4, 2017, p. 5.

[22] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the International Conference on Computer Vision (ICCV 2017), Venice, Italy*, 2017, pp. 22–29.

[23] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[24] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[25] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *European Conference on Computer Vision*. Springer, 2018, pp. 682–697.

[26] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas, "High-level multiple-UAV cinematography tools for covering outdoor events," *IEEE Transactions on Broadcasting*, 2019, accepted for publication.

[27] I. Mademlis, V. Mygdalis, C. Raptopoulou, N. Nikolaidis, N. Heise, T. Koch, J. Grunfeld, T. Wagner, A. Messina, F. Negro *et al.*, "Overview of drone cinematography for sports filming," *European Conference on Visual Media Production (CVMP), short*, 2017.

[28] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in autonomous UAV Cinematography: an overview," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[29] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous unmanned aerial vehicles filming in dynamic unstructured outdoor environments," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 147–153, 2018.

[30] I. Karakostas, I. Mademlis, N. Nikolaidis, and I. Pitas, "Shot type feasibility in autonomous UAV cinematography," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, to be presented.

[31] O. Zachariadis, V. Mygdalis, I. Mademlis, N. Nikolaidis, and I. Pitas, "2D visual tracking for sports UAV cinematography applications," *Proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2017.

[32] A. Messina, S. Metta, M. Montagnuolo, F. Negro, V. Mygdalis, I. Pitas, J. Capitán, A. Torres, S. Boyle, and D. Bull, "The future of media production through multi-drones' eyes," in *International Broadcasting Convention (IBC)*, 2018.

[33] A. Torres-González, J. Capitán, R. Cunha, A. Ollero, and I. Mademlis, "A multidrone approach for autonomous cinematography planning," *Iberian Robotics Conference (ROBOT)*, 2017.

[34] I. Mademlis, N. Nikolaidis, A. Tefas, I. Pitas, T. Wagner, and A. Messina, "Autonomous UAV cinematography: A tutorial and a formalized shot type taxonomy," *ACM Computing Surveys*, 2019, accepted for publication.

[35] D. G. Lowe, "Local feature view clustering for 3d object recognition," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[36] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3d object recognition," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. Ieee, 2010, pp. 998–1005.

[37] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. Citeseer, 2009, pp. 3212–3217.

[38] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3d object detection: A real time scalable approach," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2048–2055.

[39] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "6d object detection and next-best-view prediction in the crowd," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3583–3592.

[40] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class hough forests for 3d object detection and pose estimation," in *European Conference on Computer Vision*. Springer, 2014, pp. 462–477.

[41] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6d object pose estimation using 3d object coordinates," in *European conference on computer vision*. Springer, 2014, pp. 536–551.

[42] E. Brachmann, F. Michel, A. Krull, M. Ying Yang, S. Gumhold *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3364–3372.

[43] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1329–1335.

[44] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[45] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *IEEE International Conference on Computer Vision*, vol. 1, no. 2, 2017, p. 4.

[46] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[47] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a" siamese" time delay neural network," in *Advances in neural information processing systems*, 1994, pp.737–744.

[48] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, 2014, pp. 1988–1996.

[49] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[50] D. Q. Huynh, "Metrics for 3d rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[51] S. L. Altmann, *Rotations, quaternions, and double groups*. Courier Corporation, 2005.

[52] K. Fathian, J. P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and N. R. Gans, "Quest: A quaternion-based approach for camera motion estimation from minimal feature points," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 857–864, 2018.

[53] S. Liwicki, M.-T. Pham, S. Zafeiriou, M. Pantic, and B. Stenger, "Full-angle quaternions for robustly matching vectors of 3d rotations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 105–112.

[54] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[55] K. Perlin, "Noise hardware," *Real-Time Shading SIGGRAPH Course Notes*, 2001.

[56] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

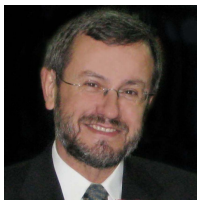
[57] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[59] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Christos Papaioannidis received his Diploma in Electrical & Computer Engineering in 2015 from the Aristotle University of Thessaloniki. He is currently pursuing his Ph.D. studies in the Artificial Intelligence & Information Analysis Laboratory in the Department of Informatics at the Aristotle University of Thessaloniki. His research interests include deep learning and image analysis.



Prof. Ioannis Pitas (IEEE fellow, IEEE Distinguished Lecturer, EURASIP fellow) received the Diploma and PhD degree in Electrical Engineering, both from the Aristotle University of Thessaloniki, Greece. Since 1994, he has been a Professor at the Department of Informatics of the same University. He served as a Visiting Professor at several Universities. His current interests are in the areas of image/video processing, intelligent digital media, machine learning, human centered interfaces, affective computing, computer vision, 3D imaging and biomedical imaging. He has published over 861 papers, contributed in 44 books in his areas of interest and edited or (co-)authored another 11 books. He has also been member of the program committee of many scientific conferences and workshops. In the past he served as Associate Editor or co-Editor of eight international journals and General or Technical Chair of four international conferences. He participated in 69 R&D projects, primarily funded by the European Union and is/was principal investigator/researcher in 41 such projects. He has 27310+ citations (Source Publish and Perish), 8216+ (Scopus) to his work and h-index 80+ (Source Publish and Perish), 44+ (Scopus).