Revisiting Fully Convolutional Geometric Features for Object 6D Pose Estimation

Jaime Corsetti
jaime.corsetti98@gmail.com

Davide Boscaini
dboscaini@fbk.eu

Fabio Poiesi
poiesi@fbk.eu

Fondazione Bruno Kessler, Italy

Abstract

Recent works on 6D object pose estimation focus on learning keypoint correspondences between images and object models, and then determine the object pose through RANSAC-based algorithms or by directly regressing the pose with end-to-end optimisations. We argue that learning point-level discriminative features is overlooked in the literature. To this end, we revisit Fully Convolutional Geometric Features (FCGF) and tailor it for object 6D pose estimation to achieve state-of-the-art performance. FCGF employs sparse convolutions and learns point-level features using a fully-convolutional network by optimising a hardest contrastive loss. We can outperform recent competitors on popular benchmarks by adopting key modifications to the loss and to the input data representations, by carefully tuning the training strategies, and by employing data augmentations suitable for the underlying problem. We carry out a thorough ablation to study the contribution of each modification. The code is available at https://github.com/jcorsetti/FCGF6D.

1. Introduction

Object 6D pose estimation is the problem of finding the Euclidean transformation (i.e. pose) of an object in a scene with respect to the camera frame [15]. This problem is important for autonomous driving [29], augmented reality [30], space docking [19], robot grasping [7], and active 3D classification [38]. The main challenges are handling occlusions, structural similarities between objects, and non-informative textures. Different benchmarks have been designed to study these challenges, such as LineMod-Occluded (LMO) [1], YCB-Video (YCBV) [40], and T-LESS [14]. LMO includes poorly-textured objects in scenarios with several occlusions. In YCBV, well-textured objects appear in scenarios with fewer occlusions but more pose variations. T-LESS includes poorly-textured and geometrically-similar objects in industrial scenarios with occasional occlusions.

Object 6D pose estimation approaches based on deep learning can be classified as *one-stage* [17, 26, 24] or *two-stage* [18, 13, 12, 39]. One-stage approaches can directly regress the object pose [17, 26, 24]. Two-stage approaches can predict 3D keypoints [13, 12] or point-level correspondences between the scene and the object [39]. Correspondences can be computed through point-level features [39]. One-stage approaches are typically more efficient than their two-stage counterpart, as they require only one inference pass. However, rotation regression is a difficult optimisation task because the rotation space is non-Euclidean and non-linear, and the definition of correct orientation is ambiguous in case of symmetric objects [34]. On the other

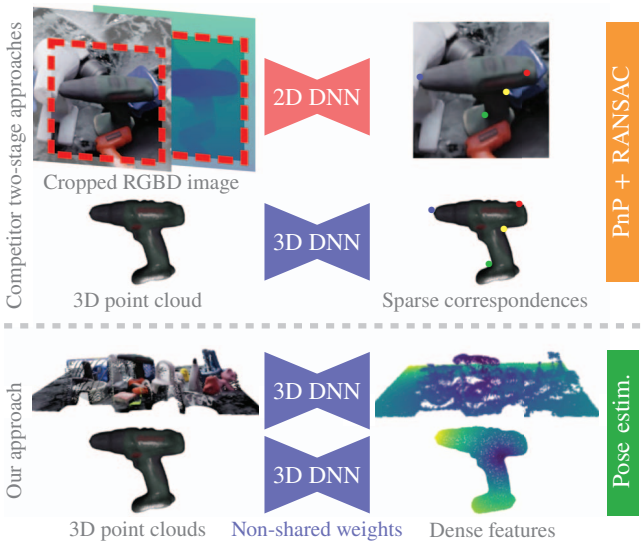

Figure 1: Top: Typically, two-stage 6D pose estimation methods process the input (RGBD image, 3D object) with different deep neural networks (2D, 3D) to learn keypoint correspondences [39], or directly predict the keypoint projections on the image [13, 12]. They also rely on detectors to crop the input image, and estimate the final pose with RANSAC-based PnP [9]. Bottom: Our method processes the whole scene and the object point clouds with 3D deep neural networks, optimises the output point-wise (dense) features by using ground-truth correspondences, and estimates the final pose with a point cloud registration algorithm.

hand, correspondence-based approaches have to be coupled with registration techniques, such as RANSAC, PnP, or least square estimation [39].

We argue that the problem of learning discriminative point-level features is overlooked in the related literature. Moreover, we believe that working at intermediate levels of representation learning, rather than regressing the pose directly, facilitates interpretability and enables us to effectively debug algorithms. Literature on representation learning for point cloud registration has made great advances [4, 32], and none of the object 6D pose estimation methods have deeply investigated the application of these techniques to the underlying problem (Fig. 1). In a landscape dominated by complex networks, our work stands as the first to comprehensively explore and quantify the benefits of this formulation with a simple yet effective solution. Our research addresses fundamental and previously unanswered questions:
i) How to learn features of heterogeneous point clouds (objects and scenes) that align in the same representation space and exhibit cross-domain generalisation (synthetic to real)?
ii) What training strategies are optimal for this approach?
iii) What degree of improvement can these strategies bring?
To answer these questions, we revisit Fully Convolutional Geometric Features (FCGF) [4] and show that its potential to achieve state-of-the-art results lies in an attentive design of data augmentations, loss negative mining, network architecture, and optimisation strategies. FCGF is designed to learn point-level features by using a fully-convolutional network optimised through a hardest contrastive loss. Compared to the original FCGF setting, our setting is asymmetric, i.e. the two input point clouds have different sizes and resolutions. Therefore, we modify the hardest contrastive loss to take into account the size of each point cloud for the mining of the hardest negatives. We use separate architectures to learn specific features for the two (heterogeneous) input data (object and scene), but unlike several state-of-the-art methods we train only a single model for all the objects of each dataset. We use specific augmentations to tackle occlusions, which are the main challenge in real-world scenarios and in the considered datasets. We name our approach FCGF6D. FCGF6D outperforms state-of-the-art methods (+3.5 ADD(S)-0.1d on LMO, +0.8 ADD-S AUC on YCBV), even when comparing with methods that train one model for each object. Our ablation study suggests that most of the performance gain is obtained thanks to our changes to the loss, the addition of the RGB information and our changes to the optimizer. In summary, our contributions are:

- We tailor FCGF for object 6D pose estimation in order to i) process entire scenes rather than cropped regions as competitors, ii) learn a single model for all objects instead of a model for each object, iii) process both photometric and geometric information with a single unified deep network model.
- A modified version of the hardest contrastive loss that is applied to heterogeneous point clouds and that considers a geometric constraint when mining the hardest negative.
- We study data augmentations that enable FCGF to improve generalisation between synthetic and real data.

2. Related work

6D pose estimation approaches can be designed to use different input data. RGB methods [18, 17, 6, 35] rely on photometric information only, while RGBD methods [13, 12, 11, 26, 39] also use range information in addition to RGB.
RGB-based 6D pose estimation. SO-Pose [6] proposes an end-to-end method that explicitly models self-occlusion maps (i.e., portions of the object that are hidden by camera orientation). It computes 2D-3D correspondences for each visible point of the object, and feeds them with self-occlusion maps to a pose regression module. ZebraPose [35] proposes a strategy to learn surface descriptors on the image, by training a neural network to predict pixel features which correspond to predefined descriptors on the object model. At inference time, it finds correspondences by similarity, and solves the PnP problem with RANSAC. The authors show that the vertex encoding process is crucial for performance improvement.
RGBD-based 6D pose estimation. PVN3D [13] extends PVNet [31] by incorporating 3D point cloud information. The core of this approach is a keypoint voting mechanism, in which for each pixel the offset to a reference keypoint is regressed. A semantic segmentation module is also used to identify the points belonging to each object in the scene. PVN3D is a two-stage method, as it passes the final correspondences to a RANSAC-based [9] algorithm for 6D pose estimation. FFB6D [12] adopts an analogous method to PVN3D [13], but introduces a novel convolutional architecture with Fusion Modules. These modules enable the model to combine photometric (RGB) and geometrical (D) features for learning a better point cloud representation. E2EK [26] proposes an end-to-end trainable method by extending FFB6D [12]. It clusters and filters the features computed by FFB6D based on confidence, and then processes them by an MLP-like network that regresses the pose. Wu et al. [39] addresses the problem of objects that are symmetric to rotation with a two-stage method. They extend FFB6D [12] by introducing a novel triplet loss based on geometric consistency. Symmetry is leveraged by considering symmetric points as positives, thus forcing them to have similar features. Feng et al. [8] proposes a method to solve a related problem. In this work, FCGF is applied to align different point clouds of objects belonging to the same category. However, the authors do not introduce task-specific modifications to FCGF, and unlike our case of application, the target object is assumed to be already segmented from the scene.

Unlike methods that employ sophisticated combinations

of deep network architectures to process RGB and depth modalities [12, 39], our approach uses deep networks based on sparse convolutions to process coloured point clouds with a single framework. Sparse convolutions are designed to process point clouds efficiently [3]. We also split the pose estimation problem into two subproblems, i.e. feature learning and point cloud registration. This allows us to evaluate the quality of the learned features by using metrics such as Feature Matching Recall [5], which fosters interpretability of our model. Unlike Wu et al. [39], we do not rely on a detector to crop the region with the candidate object before processing the point cloud with our network. Our experiments show that we can outperform the nearest competitors E2EK [26] and Wu et al. [39] by 5.7 and 1.9 ADD(S)-0.1d on the LMO dataset, respectively, without using a detector.

3. Preliminary: A review of FCGF

Input data representation. FCGF takes as input a quantised version of the original point cloud $\mathcal{X} \in \mathbb{R}^{V\times 3}$. The quantisation procedure splits the volume occupied by $\mathcal{X}$ into a grid of voxels of size Q and assigns a single representative vertex $\mathbf{x}_i \in \mathbb{R}^3$ to each voxel i. This reduction is typically computed with random sampling or by average pooling (barycenter) [3]. The resulting sparse representation is obtained by discarding voxels corresponding to a portion of the empty space and is significantly more efficient in terms of memory utilisation.

Feature extractor. The fully-convolutional feature extractor $\mathbf{\Phi}_\Theta$ is a parametric function with learnable parameters Θ designed as a UNet [33]. Given $\mathbf{x}_i$, $\mathbf{\Phi}_\Theta$ produces a F-dimensional feature vector defined as $\mathbf{\Phi}_\Theta(\mathbf{x}_i) = \mathbf{f}_i \in \mathbb{R}^F$. FCGF processes pairs of point clouds using a Siamese approach, i.e. feature extractors with shared weights. FCGF is implemented in PyTorch using Minkowski engine [3].

Hardest contrastive loss. The hardest contrastive (HC) loss is defined as $\ell_{\text{HC}} = \lambda_P \ell_P + \lambda_N \ell_N$, where ℓ_P promotes similarity between features of positive samples, ℓ_N promotes dissimilarity between features of negative samples, and λ_P, λ_N are hyperparameters. Given a pair of 3D scenes $(\mathcal{X}_1, \mathcal{X}_2)$ as input, the set of positive pairs is defined as $\mathcal{P} = \{(i,j) : \mathbf{x}_i \in \mathcal{X}_1, \mathbf{x}_j \in \mathcal{X}_2, \phi(\mathbf{x}_i) = \mathbf{x}_j\}$, where $\phi : \mathcal{X}_1 \to \mathcal{X}_2$ is a correspondence mapping between $\mathcal{X}_1$ and $\mathcal{X}_2$ voxels. ℓ_P is defined as

$$\ell_P = \sum_{(i,j)\in\mathcal{P}} \frac{1}{|\mathcal{P}|} \left(\|\mathbf{f}_i - \mathbf{f}_j\| - \mu_P \right)_+^2, \tag{1}$$

where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$, μ_P is a positive margin to overcome overfitting [25], and $(\cdot)_+ = \max(0, \cdot)$. For each pair $(i,j) \in \mathcal{P}$, two sets of candidate negatives are defined as $\mathcal{N}_i = \{k \text{ s.t. } \mathbf{x}_k \in \mathcal{X}_1, k \neq i\}$, $\mathcal{N}_j = \{k \text{ s.t. } \mathbf{x}_k \in \mathcal{X}_2, k \neq j\}$. Computing $\mathcal{N}_i, \mathcal{N}_j$ scales quadratically with the mini-batch size, therefore random subsets of $\mathcal{N}_i$ and $\mathcal{N}_j$ with fixed cardinalities are instead used in practice. ℓ_N is defined as

$$\begin{aligned} \ell_N = \sum_{(i,j)\in\mathcal{P}} & \frac{1}{2|\mathcal{P}_i|} \left(\mu_N - \min_{k\in\mathcal{N}_i} \|\mathbf{f}_i - \mathbf{f}_k\| \right)_+^2 \\ & + \frac{1}{2|\mathcal{P}_j|} \left(\mu_N - \min_{k\in\mathcal{N}_j} \|\mathbf{f}_j - \mathbf{f}_k\| \right)_+^2, \end{aligned} \tag{2}$$

where $|\mathcal{P}_i|, |\mathcal{P}_j|$ are the numbers of valid negatives mined from the first and second term, respectively. Unlike metric learning losses that randomly mine a certain number of negatives from $\mathcal{N}_i, \mathcal{N}_j$ [10, 37], the HC loss mines the most similar features within a batch, i.e. the hardest negatives.

4. Tailoring FCGF for 6D pose estimation

In this section, we describe how we modified FCGF. We focus on manipulating heterogeneous representations of input data, improving the HC loss, and modernising the training strategy. Fig. 2 shows the block diagram of FCGF6D.

4.1. Input data

Heterogeneous representations. FCGF was designed for scene registration, where its input data is 3D scan pairs of the same scene captured from different viewpoints. Therefore, their input data belongs to the same distribution, i.e. real-world data captured with the same LiDAR sensor. This is why authors in [4] use a Siamese approach. Unlike FCGF, our input data is heterogeneous, therefore we process it with two independent deep networks. Formally, given an object O and a scene S, the input of our pipeline is the pair $(\mathcal{M}_O, \mathcal{I}_S)$, where $\mathcal{M}_O$ is a textured 3D model of O and $\mathcal{I}_S$ is an RGBD capture of S from a viewpoint. We transform $(\mathcal{M}_O, \mathcal{I}_S)$ into a pair of point clouds. For O, we produce a point cloud $\mathcal{X}_O \in \mathbb{R}^{V_O\times 6}$ by sampling V_O vertices on the triangular faces of $\mathcal{M}_O$ and extracting the corresponding RGB colours from its texture. For S, we use the intrinsic parameters of the RGBD sensor to map $\mathcal{I}_S$ into a coloured point cloud and sample V_S points from it. Let $\mathcal{X}_S \in \mathbb{R}^{V_S\times 6}$ be the point cloud of S. We quantise $\mathcal{X}_O$ and $\mathcal{X}_S$ by a factor Q and process the pair with two networks implemented with Minkowski engine [3]. V_O, V_S, and Q are hyperparameters.

Processing geometric and photometric data. Minkowski engine [3] is designed to process optional input features in addition to the 3D coordinate of each point. However, authors in [4] show that, in the context of scene registration, adding the photometric information associated to each point leads to overfitting. We found instead that this addition significantly improves the performance. Colour information helps in i) discriminating objects of different categories but with similar geometric shape (e.g. pudding box and gelatin box in YCBV [40]), and ii) selecting the correct pose of symmetric objects among the set of geometrically-equivalent ones (i.e.

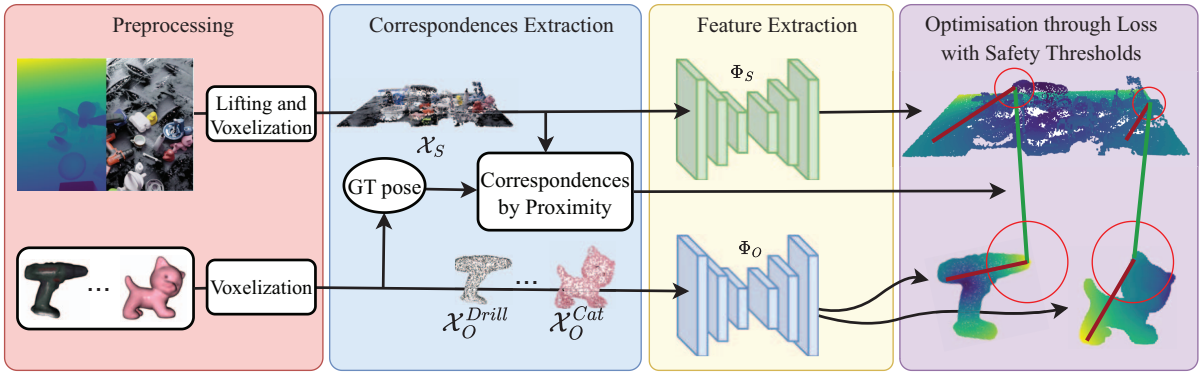

Figure 2: FCGF6D training pipeline consists of four logical parts. Given a scene S and an object O we take as input the pair $(\mathcal{I}_S, \mathcal{M}_O)$. In the first part, we compute 3D point cloud representations $(\mathcal{X}_S, \mathcal{X}_O)$ of $(\mathcal{I}_S, \mathcal{M}_O)$, where $\mathcal{X}_S$ is obtained by lifting $\mathcal{I}_S$ using the intrinsic parameters of the camera that acquired it, and then quantise them. In the second part, we mine positives by computing the correspondences between $\mathcal{X}_S$ and $\widetilde{\mathcal{X}}_O = \mathbf{R}_O\mathcal{X}_O + \mathbf{t}_O$, where $\mathbf{R}_O, \mathbf{t}_O$ is the ground-truth 6D pose of O. In the third part, we perform point-wise feature extraction with two independent UNets Φ_S, Φ_O. In the fourth part, the hardest contrastive loss with safety thresholds is applied to guide the feature learning process.

the 6D pose of a box or a can cannot be uniquely defined unless we consider their texture patterns).

4.2. Loss function

Positive mining. We define $\mathcal{P}$ (Eq. 1) as the set of valid correspondences between $\mathcal{X}_O$ and $\mathcal{X}_S$. Let $(\mathbf{R}_O, \mathbf{t}_O)$ be O ground-truth 6D pose in S and $\widetilde{\mathcal{X}}_O = \mathbf{R}_O\mathcal{X}_O + \mathbf{t}_O$ be the rigidly transformed version of $\mathcal{X}_O$ into the reference frame of $\mathcal{X}_S$. We compute all the correspondences by searching for each point of $\widetilde{\mathcal{X}}_O$ its nearest neighbouring point in $\mathcal{X}_S$. Due to occlusions with other objects and/or self-occlusions, some of the correspondences may be spurious, e.g. associating points of different surfaces. Therefore, we consider a correspondence valid if the distance between $\tilde{\mathbf{x}}_i \in \widetilde{\mathcal{X}}_O$ and $\mathbf{x}_j \in \mathcal{X}_S$ is less than a threshold τ_P and if the other points on the scene are farther away, i.e. $(i, j) \in \mathcal{P} \Leftrightarrow \|\tilde{\mathbf{x}}_i - \mathbf{x}_j\| < \tau_P$ and $\|\tilde{\mathbf{x}}_i - \mathbf{x}_j\| < \|\tilde{\mathbf{x}}_i - \mathbf{x}_k\|$ for every $k = 1, \dots, V_S$.

Negative mining. We experienced that mining the hardest negatives from the negative sets $\mathcal{N}_i$, $\mathcal{N}_j$ (Eq. 2) can lead to loss instability and collapsing. This occurs because the hardest negative in $\mathcal{N}_i = \{k : \mathbf{x}_k \in \mathcal{X}_O, k \neq i\}$, i.e. the sample with the closest feature to $\mathbf{f}_i \in \mathbb{R}^F$, is likely to be a point spatially close to $\mathbf{x}_i \in \mathcal{X}_O$, because their local geometric structure is nearly the same. Hence, Eq. 2 tries to enforce features corresponding to the same local geometric structure to be distant from each other. This problem can be mitigated by replacing $\mathcal{N}_i, \mathcal{N}_j$ in Eq. 2 with $\widetilde{\mathcal{N}_i} = \{k : \mathbf{x}_k \in \mathcal{X}_O, \|\mathbf{x}_k - \mathbf{x}_i\| > \tau_{NO}\}$ and $\widetilde{\mathcal{N}_j} = \{k : \mathbf{x}_k \in \mathcal{X}_S, \|\mathbf{x}_k - \mathbf{x}_j\| > \tau_{NO}\}$, where τ_{NO} is a safety threshold, i.e. the radius of spheres on object and on scene where mining is forbidden.

The choice of τ_{NO} is key because it determines which points on the point clouds can be used for negative mining. We found beneficial to choose τ_{NO} as a function of the dimension of the input object. Given $\mathcal{X}_O$, we define its diameter as D_O, and set $\tau_{NO} = t_{\text{scale}} D_O$. In Fig. 3, we illustrate the safety thresholds. In this way, we can maintain a good quantity of negatives while avoiding the mining of spurious hardest negatives. Using different thresholds for the object and the scene points clouds underperformed our final choice. Therefore, our loss is defined as

$$\ell_{\text{HC}} = \lambda_P \ell_P + \lambda_{NO} \ell_{NO} + \lambda_{NS} \ell_{NS},$$

where λ_P, λ_{NO} and λ_{NS} are weight factors. τ_P, t_{scale}, λ_P, λ_{NO}, and λ_{NS} are hyperparameters.

4.3. Training strategy

Data augmentation. FCGF combines scaling and rotation augmentations to enhance feature robustness against variations in camera pose [4]. These are effective in the context of point cloud registration, but in our specific scenario, the point cloud of the objects always belongs to a known set. Avoiding these augmentations helps the deep network in learning specialised features for each object. Our data augmentations consist of the following:

(i) Point re-sampling of O and S, i.e. unlike FCGF, we randomly downsample point clouds at each epoch to mitigate overfitting. This allows the model to be more robust to depth acquisition noise; (ii) Colour jittering on O, i.e. we randomly perturb brightness, contrast, saturation, and hue of O; (iii) Random erasing on S, i.e. unlike FCGF, we simulate occlusions at training time. For each point of $\widetilde{\mathcal{X}}_O$ we compute its nearest neighbour in $\mathcal{X}_S$ and randomly select a point on $\mathcal{X}_S$ within such correspondence set. We then erase all the points that fall within a distance threshold ρ from it. This allows the model to be more robust to occlusions in the input scene.

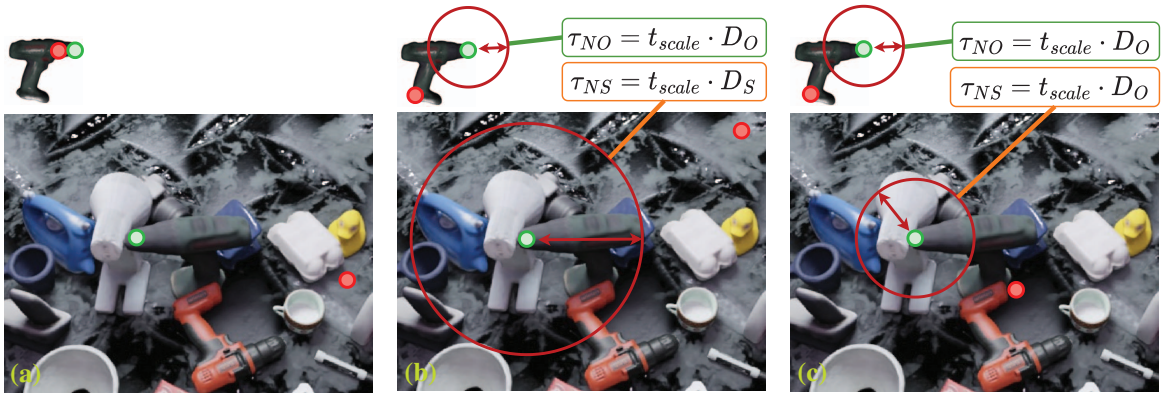

Figure 3: Examples of different mining strategies. (a) Hardest contrastive loss as proposed in FCGF: no constraints are enforced on the location of the hardest negative (red point) with respect to the correspondent point (green point). (b) A vanilla choice of the safety thresholds: the radii τ_{NS}, τ_{NO} are proportional to the diameters D_S, D_O of the respective point clouds. (c) Our choice: the value of the thresholds is proportional to the diameter of the object, i.e. $\tau_{NS} = \tau_{NO}$.

Optimisation techniques. FCGF uses an SGD optimiser with an initial learning rate $\text{lr}_{\text{init}} = 10^{-1}$ decreased during training with an exponential scheduler with $\gamma = 0.99$. In our setting, these hyperaparameters do not lead to convergence. Instead, we set $\text{lr}_{\text{init}} = 10^{-3}$. We experiment with Adam [21] and AdamW [28], and notice improvements in both cases. We also switch to a Cosine Annealing scheduler [27] that lowers the learning rate from 10^{-3} to 10^{-4} across the epochs.

5. Experiments

5.1. Datasets

We evaluate FCGF6D on the LineMod-Occluded (LMO) [1] and the YCB-Video (YCBV) [40] datasets.

LMO [1] contains RGBD images of real scenes with different configurations of objects placed on a table. It provides the ground-truth 6D pose of eight of these objects, which are always present in the scene. Objects are poorly textured, of varying dimensions and placed in a cluttered scene, featuring a variety of lightning conditions. We use the original test set of 1,213 real images, while for the training set the works we use as comparison use different combinations of synthetic and real images: the methods they use to generate the synthetic images and the number of samples for each type are not always clearly defined [13, 12, 26, 39]. Differently, we only use the Photo Realistic Rendering (PBR) set of 50,000 synthetic images provided by the BOP challenge [16] as it contains a large variety of pose configurations. Following [13], we adopt an hole filling algorithm [22] to improve the depth quality on both training and test images.

YCBV [40] contains RGBD images of real scenes with different configurations of 21 objects taken from the YCB dataset [2]. Objects have similar geometry (e.g. boxes and cans) and are placed in various poses (e.g. some objects are placed on top of others). Unlike LMO, the objects are placed in different contexts. We use the original test set of 20,738 real images. As for LMO, state-of-the-art methods use different combinations of synthetic and real data [13, 12, 26, 39]. For training, we choose 4,000 synthetic, 4,000 real, and 4,000 PBR images provided by the BOP challenge [16] because we found that using only the PBR images leads to unsatisfactory results. Also for YCBV we adopt a hole filling algorithm [22] on both train and test depth images as done in [13].

5.2. Implementation details

LMO setting. Experiments on LMO share the following hyperparameters. The input pair (O, S) is first sampled to $V_O = 4{,}000$ and $V_S = 50{,}000$ points, respectively, and then quantised with a step of $Q = 2$mm. As feature extractor we use a MinkUNet34 [3] with output dimension $F = 32$. The correspondence estimation threshold used for the positive mining is $\tau_P = 4$mm, and the maximum number of correspondences extracted is set to 1,000. The safety threshold τ_{NO} is defined proportionally to the object O diameter by setting $t_{\text{scale}} = 0.1$ (see Fig. 3). The hardest negative mining on $\mathcal{X}_O$ is performed in $\widetilde{\mathcal{N}_i}$. When mining the hardest negatives on $\mathcal{X}_S$, instead of considering the full candidates set $\widetilde{\mathcal{N}_j}$ we randomly sample 10,000 points from it to reduce the spatial complexity. HC loss margins are set as $\mu_P = 0.1$, $\mu_N = 10$, and coefficients are set to $\lambda_P = 1$, $\lambda_{NO} = 0.6$, and $\lambda_{NS} = 0.4$. The feature extractor is trained on 50,000 PBR images for 12 epochs. The pose is obtained by using the TEASER++ [41] algorithm.

YCBV setting. Experiments on YCBV share the same LMO hyperparameters except in the following cases. We set $V_S = 20{,}000$, as we found that it works on par with the original V_S of LMO. We believe this happens because YCBV objects are less occluded and their geometries are less complex than

Table 1: Comparison of RGB and RGBD methods performance on LMO [1] evaluated in terms of ADD(S)-0.1d. Key: *: symmetric object, DNNs: number of Deep Neural Networks used, n.a.: information not available, Det: object detections are used as prior, Seg: object segmentation masks are used as prior, **bold**: best result, underline: second best result.

Input	Method	DNNs	Prior	Ape	Can	Cat	Drill	Duck	Eggbox*	Glue*	Holepuncher	Avg
RGB	SO-Pose [6]	1	Det	48.4	85.8	32.7	77.4	48.9	52.4	78.3	75.3	62.3
	ZebraPose [35]	8	Det	57.9	95.0	60.6	94.8	64.5	70.9	88.7	83.0	76.9
RGBD	PVN3D [13]	1	–	33.9	88.6	39.1	78.4	41.9	80.9	68.1	74.7	63.2
	PR-GCN [42]	n.a.	Det	40.2	76.2	57.0	82.3	30.0	68.2	67.0	97.2	65.0
	FFB6D [12]	1	–	47.2	85.2	45.7	81.4	53.9	70.2	60.1	85.9	66.2
	DCL-Net [23]	n.a.	Det	56.7	80.2	48.1	81.4	44.6	83.6	79.1	91.3	70.6
	E2EK [26]	8	Seg	61.0	95.4	50.8	94.5	59.6	55.7	78.3	91.4	73.3
	Wu et al. [39]	8	Det	66.1	97.4	70.7	95.4	70.1	61.2	59.8	95.7	77.1
	FCGF6D (ours)	1	–	65.4	96.7	64.8	97.8	71.7	54.1	83.2	97.9	79.0
	FCGF6D (ours)	1	Det	63.6	94.8	63.4	97.4	73.4	74.6	80.4	97.3	**80.6**

LMO objects. As feature extractor we use a MinkUNet50 model [3], trained on 12,000 mixed images for 110 epochs. The pose is obtained with a RANSAC-based algorithm from Open3D [43]. Experimentally, on YCBV we found that RANSAC yields better results than TEASER++. We believe that this happens because TEASER++ is heavily based on correspondences [41] and for YCBV we use a lower resolution for the scene compared to LMO, which in turn reduces the number of correspondences.

5.3. Evaluation metrics

We use the ADD and ADD-S metrics that are defined as

$$\text{ADD} = \frac{1}{V_O} \sum_{\mathbf{x} \in \mathcal{X}_O} \left\| (\mathbf{R}\mathbf{x} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}}) \right\|,$$

$$\text{ADD-S} = \frac{1}{V_O} \sum_{\mathbf{x}_1 \in \mathcal{X}_O} \min_{\mathbf{x}_2 \in \mathcal{X}_O} \left\| (\mathbf{R}\mathbf{x}_1 + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x}_2 + \hat{\mathbf{t}}) \right\|,$$

where $\mathbf{R}, \mathbf{t}$ and $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ are the translation and rotation components of the predicted and the ground-truth poses of $\mathcal{X}_O \in \mathbb{R}^{V_O \times 3}$, respectively. ADD(S) computes the ADD for non-symmetric objects and the ADD-S for symmetric ones. Performance on LMO is assessed in term of the ADD(S)-0.1d metric [13, 12, 26, 39], which computes the percentage of ADD(S) errors lower than 10% of the object diameter [15]. Performance on YCBV is assessed in term of the ADD-S AUC metric [40, 13, 12]. The area-under-the-curve (AUC) of ADD-S is obtained by computing the cumulative percentage of ADD-S errors lower than a threshold varying from 1mm to 100mm. Note that in ADD(S)-0.1d the success thresholds are relative to the object diameters, while in ADD-S AUC they are absolute.

5.4. Quantitative results

Tab. 1 reports the results on LMO [1] in term of ADD(S)-0.1d: for completeness we added the two best performing RGB methods (top), while the other ones are RGBD methods (bottom). As reported in the Prior column, most methods rely on additional priors, either in the form of object detections (Det) or of object segmentation masks (Seg). FCGF6D outperforms all the other methods by a large margin without using any prior (penultimate row): it outperforms Wu et al. by 1.9%, E2EK by 5.7%, DCL-Net by 8.4%, FFB6D by 12.8%, PR-GCN by 14.0%, and PVN3D by 15.8%. Note that Wu et al. [39] and E2EK [26] train a different deep neural network for each object (DNNs column), whereas we train only a single deep neural network, saving learning parameters and training time. Moreover, when we use the object detections obtained with YOLOv8 [20] (last row), the performance of FCGF6D further improves, outperforming Wu et al. by 3.5%, E2EK by 7.3%, and all the other methods by more than 10.0%. Note that detectors are prone to errors: when detections are wrong, the object pose will be wrong too. We can observe that the detector is more effective with Duck and Eggbox. The first is a particularly small object, therefore more likely to be occluded. The second undergoes frequent occlusions (other objects are on top of it in several images), thus making localisation difficult without a detector. To further understand the negative impact of the detector, we compute the percentage of poses which are wrong when we use detections and correct when we do not use detections. For Ape, Can and Glue, this percentage is 3.3%, 1.7%, and 5.1%, respectively. Please refer to the Supplementary Material for a comprehensive analysis of the detector impact.

Tab. 2 reports the results on YCBV [40] in ADD-S AUC compared with other RGBD-based methods. The row Prior indicates eventual additional priors used by each method. The default configuration of FCGF6D does not require any input prior and uses a deep neural network for all the objects. FCGF6D outperforms recent competitors that do not use input priors: it outperforms FFB6D by 0.8% and PVN3D by 1.7%. E2EK [26] and Wu et al. [39] instead consider input priors in the form of object segmentation masks and object detections, respectively, and train a model for each object (DNNs row). When we use input priors in the form of detections, FCGF6D outperforms E2EK by 2.4% and slightly underperforms Wu et al. by −0.6%. We also observe that, thanks to multi-scale representation provided by the UNet, we obtain good performance also on symmmetric objects

Table 2: Performance of RGBD methods on YCBV [40] evaluated in ADD-S AUC. Key: *: symmetric object, DNNs: number of Deep Neural Networks used, Det: object detections are used as prior, Seg: object segmentation masks are used as prior, **bold**: best result, underline: second best result.

Method	PVN3D [13]	FFB6D [12]	FCGF6D (ours)	E2EK [26]	Wu et al. [39]	FCGF6D (ours)
DNNs	1	1	1	21	21	1
Prior	–	–	–	Seg	Det	Det
master chef can	80.5	80.6	96.1	79.6	100.0	96.3
cracker box	94.8	94.6	96.4	95.1	98.8	96.7
sugar box	96.3	96.6	98.1	96.7	100.0	98.1
tomato soup can	88.5	89.6	93.1	89.8	97.5	95.8
mustard bottle	96.2	97.0	98.3	96.5	100.0	98.3
tuna fish can	89.3	88.9	82.8	90.7	99.9	97.6
pudding box	95.7	94.6	95.2	96.9	100.0	97.3
gelatin box	96.1	96.9	98.7	97.5	100.0	98.7
potted meat can	88.6	88.1	79.9	90.8	84.1	89.8
banana	93.7	94.9	98.3	94.4	100.0	98.3
pitcher base	96.5	96.9	97.9	95.6	100.0	97.9
bleach cleanser	93.2	94.8	95.9	94.0	99.9	96.7
bowl*	90.2	96.3	97.3	96.0	94.5	98.2
mug	95.4	94.2	97.4	95.3	100.0	97.7
power drill	95.1	95.9	98.2	96.6	100.0	98.2
wood block*	90.4	92.6	95.2	93.8	98.0	96.4
scissors	92.7	95.7	93.9	97.9	100.0	95.9
large marker	91.8	89.1	97.5	95.0	99.9	98.3
large clamp*	93.6	96.8	80.6	97.2	91.1	93.8
extra large clamp*	88.4	96.0	77.4	96.7	81.0	94.7
foam brick*	96.8	97.3	94.6	97.2	99.8	97.6
Avg	91.8	92.7	**93.5**	94.4	**97.4**	96.8

without the need of specific techniques to handle symmetry. Note that we employed detections in both Tabs. 1&2 to illustrate their potential use in improving registration efficacy, though not obligatory. Specifically in Tab. 2, when we compare with methods based on the same assumptions as ours, FCGF6D achieves state-of-the-art performance, see comparison with PVN3D [13] and FFB6D [12]. When we compare with methods that use 21 models instead of 1 (as ours), we fall slightly behind the best (see comparison with E2EK [26] and Wu et al. [39]).

5.5. Qualitative results

Fig. 4 shows some examples of successes and failures on the test set of LMO dataset. The upper row shows the ground-truth poses, and the bottom one shows the poses predicted by our model. Note how FCGF6D is capable of estimating the correct pose even in case of partial objects (i.e. the glue in the first image). However, our model fails in case of partial objects with ambiguities (the duck in the second image), or of atypical occlusions (the eggbox in the second image: the training set do not contain this degree of occlusions).

Fig. 5 shows some examples of successes and failures on the test set of YCBV. FCGF6D appears prone to rotation errors (the large clamp in the first image), especially in case of partially occluded objects (the bleach cleanser in the second image). However, the poses are generally accurate.

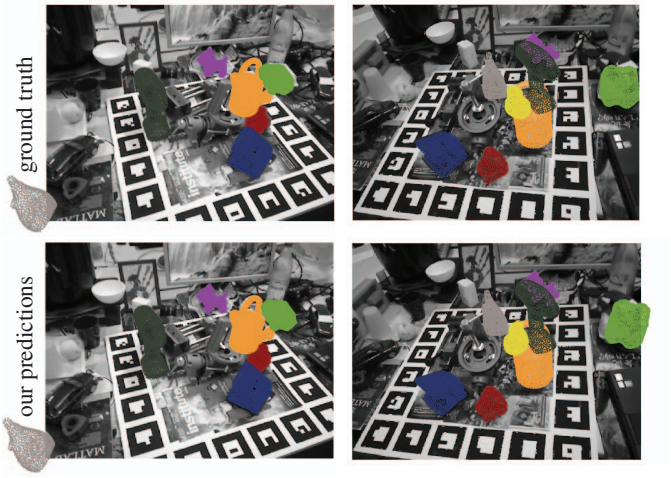

Figure 4: Qualitative results on LMO [1]. Colour key: • Ape, • Can, • Cat, • Drill, • Duck, • Eggbox, • Glue, • Holepuncher.

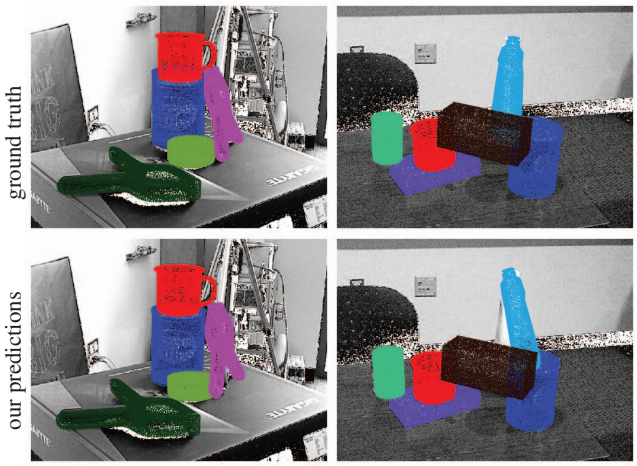

Figure 5: Qualitative results on YCBV [40]. Colour key: • master chef can, • sugar box, • tomato soup can, • tuna fish can, • bleach cleanser, • mug, • wood block, • large clamp, • extra large clamp.

5.6. Ablation study

We conduct an ablation study on the Drill object of the LMO dataset by training FCGF6D for five epochs. We choose the closest setting to FCGF as baseline: no safety threshold in the loss, shared network weights, no RGB information, SGD optimiser with $\text{lr}_{\text{init}} = 10^{-3}$, exponential scheduler with $\gamma = 0.99$. We perform a single experiment for each added component to assess their individual contribution. As metrics, we use ADD(S), Relative Rotation Error (RRE), Relative Translational Error (RTE), and Feature Matching Recall (FMR) [4, 32]. RRE and RTE show how the two pose components (rotation and translation) are

Table 3: Ablation study on the Drill object of LMO. Performance is compared in RRE [radians] and RTE [cm] errors, FMR and ADD(S)-0.1d (shortened to ADD) scores. Δ shows the improvement of each contribution in terms of ADD(S)-0.1d with respect to the previous row.

	Improvements	RRE ↓	RTE ↓	FMR ↑	ADD ↑	Δ
	Baseline	2.2	9.6	0	0.2	–
Loss	+ $\tau_{NS} = 0.1D_S$	1.8	12.2	0	0.4	+0.2
	+ $\tau_{NS} = 0.1D_O$	1.1	5.3	0.2	18.2	+17.8
Arch.	+ Independent weights	1.2	3.7	0	29.1	+10.9
	+ Add RGB information	0.6	2.2	38.5	63.3	+34.2
Aug.	+ Colour augmentation	0.6	2.2	32.0	65.4	+2.1
	+ Random erasing	0.3	1.8	78.4	75.6	+10.2
Optim.	+ SGD → Adam	0.1	1.1	93.4	95.8	+20.2
	+ Adam → AdamW	0.1	0.9	93.9	96.4	+0.6
	+ Exp → Cosine	0.1	0.9	93.6	96.6	+0.2

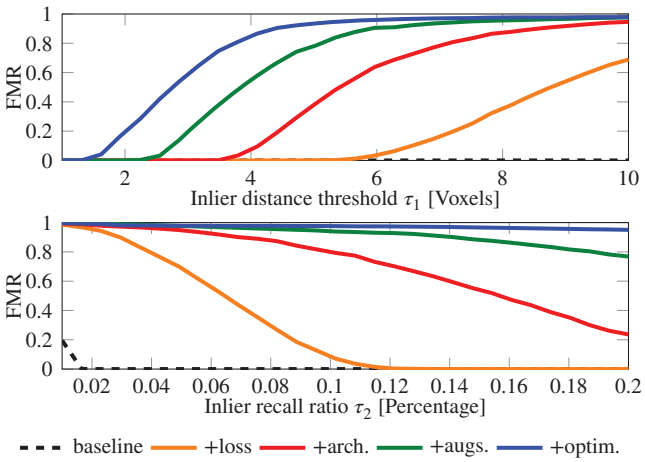

Figure 6: Feature Matching Recall (FMR) as a function of τ_1 and τ_2. When varying τ_1 (top) we set τ_2=5%, and when varying τ_2 (bottom) we set τ_1=10 voxels.

affected. FMR indirectly measures the number of iterations required by a registration algorithm, e.g. RANSAC, to estimate the transformation between two point clouds. We set the inlier distance threshold as $\tau_1 = 5$ voxels, and the inlier recall ratio as $\tau_2 = 5\%$.

Tab. 3 shows that the largest contributions in ADD(S)-0.1d are: introducing the safety threshold in the loss (+17.8), adding RGB information (+34.2), and adopting Adam optimiser (+20.2). We also note that the gain in ADD(S)-0.1d is not always consistent with the FMR: when RGB augmentation is added, there is a gain in ADD(S)-0.1d of 2.1, but the FMR drops by 6.5. A more detailed analysis of FMR with different values of τ_1 and τ_2 is shown in Fig. 6.

5.7. Training and inference time

The training time is about one week for each dataset using two NVIDIA A40 GPUs. Tab. 4 reports the comparison of the number of parameters, inference GPU memory footprint, and inference time (using a GeForce RTX 3050 GPU) on YCBV. We were unable to test E2EK [26] as the code is unavailable, whereas we used the authors' original code for the other papers. FCGF6D has a significantly smaller memory footprint than the main competitors, and the inference time is comparable. In a scenario where multiple objects are expected, our closest competitor [39] uses a different model for each object, thereby requiring more memory. Our method requires less memory because we train only a single model. Note that using the whole scene as input is advantageous in a practical scenario where N instances of the same object are present. Here, we need a single forward pass, followed by N registrations. Instead, methods that rely on image crops [42, 23, 39] require a forward pass for each instance.

Table 4: Inference time and memory footprint. Time is for a single image, and includes network inference (inf.) and registration (reg.) times. N is the number of trained models.

Method	DNNs	Params [M]	Memory [GB]	Time [ms] (inf.+reg.)
PVN3D [13]	1	38.6	3.17	417 (154 + 263)
FFB6D [12]	1	33.8	2.46	285 (146 + 139)
Wu et al. [39]	N	23.8×N	2.04×N	144 (143 + 1)
Ours	1	63.5	1.3	156 (118 + 38)

6. Conclusions

We revisited the Fully Convolutional Geometric Feature (FCGF) approach to tackle the problem of object 6D pose estimation. FCGF uses sparse convolutions to learn point-wise features while optimising a hardest contrastive loss. Key modifications to the loss, input data representations, training strategies, and data augmentations to FCGF enabled us to outperform competitors on popular benchmarks. A thorough analysis is conducted to study the contribution of each modification to achieve state-of-the-art performance. Future research directions include the application of our approach to generalisable 6D pose estimation [36].

Limitations. Minkowski engine is computational efficient but has a large memory footprint at training time. We mitigated this by downsampling the scene point cloud and by adopting quantisation. It would be interesting to understand how not to lose the input point cloud resolution while maintaining a modest memory footprint.

Acknowledgements

We are grateful to Andrea Caraffa for his support with the computation of the detection priors and to Nicola Saljoughi for his contributions during the early stage of the project.

This work was supported by the European Union's Horizon Europe research and innovation programme under grant agreement No 101058589 (AI-PRISM), and by the PNRR project FAIR - Future AI Research (PE00000013), under the NRRP MUR program funded by the NextGenerationEU.

References

[1] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D object pose estimation using 3D object coordinates. In *ECCV*, 2014. 1, 5, 6, 7

[2] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A.M. Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *ICAR*, 2015. 5

[3] C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 3, 5, 6

[4] C. Choy, J. Park, and V. Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 2, 3, 4, 7

[5] H. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3D point matching. In *CVPR*, 2018. 3

[6] Y. Di, F. Manhardt, G. Wang, X. Ji, N. Navab, and F. Tombari. SO-Pose: Exploiting Self-Occlusion for Direct 6D Pose Estimation. In *ICCV*, 2021. 2, 6

[7] G. Du, K. Wang, S. Lian, and K. Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54, 2021. 1

[8] Qiaojun Feng and Nikolay Atanasov. Fully convolutional geometric features for category-level object alignment. 2020. 2

[9] M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24, 1981. 1, 2

[10] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 3

[11] R.L. Haugaard and A.G. Buch. SurfEmb: Dense and Continuous Correspondence Distributions for Object Pose Estimation with Learnt Surface Embeddings. In *CVPR*, 2022. 2

[12] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun. FFB6D: A Full Flow Bidirectional Fusion Network for 6D Pose Estimation. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7, 8

[13] Y. He, W. Sun, H. Huang, J. Liu, H. Fan, and J. Sun. PVN3D: A Deep Point-Wise 3D Keypoints Voting Network for 6DoF Pose Estimation. In *CVPR*, 2020. 1, 2, 5, 6, 7, 8

[14] T. Hodan, P. Haluza, S. Obdrzalek, J. Matas, M. Lourakis, and X. Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. *WACV*, 2017. 1

[15] T. Hodan, J. Matas, and S. Obdrzalek. On evaluation of 6D object pose estimation. In *ECCV*, 2016. 1, 6

[16] T. Hodan, M. Sundermeyer, B. Drost, Y. Labbe, E. Brachmann, F. Michel, C. Rother, and J. Matas. BOP challenge 2020 on 6D object localization. *ECCV Workshops*, 2020. 5

[17] Y. Hu, P. Fua, W. Wang, and M. Salzmann. Single-stage 6d object pose estimation. In *CVPR*, 2020. 1, 2

[18] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann. Segmentation-driven 6d object pose estimation. In *CVPR*, 2019. 1, 2

[19] Y. Hu, S. Speierer, W. Jakob, P. Fua, and M. Salzmann. Wide-Depth-Range 6D Object Pose Estimation in Space. In *CVPR*, 2021. 1

[20] G. Jocher, A. Chaurasia, and J. Qiu. YOLO by Ultralytics, 2023. 6

[21] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. 5

[22] J. Ku, A. Harakeh, and S.L. Waslander. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. In *Conference on Computer and Robot Vision*, 2018. 5

[23] H. Li, J. Lin, and K. Jia. DCL-Net: Deep Correspondence Learning Network for 6D Pose Estimation. In *ECCV*, 2022. 6, 8

[24] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox. Deepim: Deep iterative matching for 6d pose estimation. In *ECCV*, 2018. 1

[25] J. Lin, O. Morere, V. Chandrasekhar, A. Veillard, and H. Goh. Deephash: Getting regularization, depth and fine-tuning right. *arXiv:1501.04711*, 2015. 3

[26] S. Lin, Z. Wang, Y. Ling, Y. Tao, and C. Yang. E2EK: End-to-End Regression Network Based on Keypoint for 6D Pose Estimation. *RAL*, 7, 2022. 1, 2, 3, 5, 6, 7, 8

[27] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[28] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 5

[29] F. Manhardt, W. Kehl, and A. Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *CVPR*, 2019. 1

[30] E. Marchand, H. Uchiyama, and F. Spindler. Pose estimation for augmented reality: a hands-on survey. *Visualization and Computer Graphics*, 22, 2015. 1

[31] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao. PVNet: Pixel-wise voting network for 6dof pose estimation. In *CVPR*, 2019. 2

[32] F. Poiesi and D. Boscaini. Learning general and distinctive 3D local deep descriptors for point cloud registration. *TPAMI*, 2022. 2, 7

[33] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI*, 2015. 3

[34] A. Saxena, J. Driemeyer, and A. Ng. Learning 3-d object orientation from images. In *ICRA*, 2009. 1

[35] Y. Su, M. Saleh, T. Fetzer, J. Rambach, N. Navab, B. Busam, D. Stricker, and F. Tombari. ZebraPose: Coarse to Fine Surface Encoding for 6DoF Object Pose Estimation. In *CVPR*, 2022. 2, 6

[36] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou. Onepose: One-shot object pose estimation without cad models. In *CVPR*, 2022. 8

[37] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 3

[38] Y. Wang, M. Carletti, F. Setti, M. Cristani, and A. Del Bue. Active 3D Classification of Multiple Objects in Cluttered Scenes. In *ICCVW*, 2019. 1

[39] C. Wu, L. Chen, S. Wang, H. Yang, and J. Jiang. Geometric-aware Dense Matching Network for 6D Pose Estimation of Objects from RGB-D Images. *Pattern Recognition*, 2023. 1, 2, 3, 5, 6, 7, 8

[40] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In *RSS*, 2018. 1, 3, 5, 6, 7

[41] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *TRO*, 37, 2020. 5, 6

[42] G. Zhou, H. Wang, J. Chen, and D. Huang. Pr-gcn: A deep graph convolutional network with point refinement for 6d pose estimation. In *ICCV*, 2021. 6, 8

[43] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 6