# Back to 3D: Few-Shot 3D Keypoint Detection with Back-Projected 2D Features

## Supplementary Material

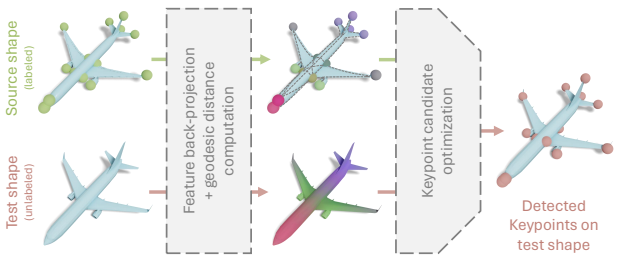


Figure 10. Pipeline of B2-3D. Given one or a several labeled source shapes and an unlabeled *test shape*, we first back-project features for ground-truth keypoints on the source shapes and candidate locations sampled from the surface of the test shape. In addition, we compute the pairwise geodesic distances between the keypoints. With this information at hand we then employ our optimization module to detect the keypoints on the test shape. Intuitively, our optimization uses the back-projected features as *first order* similarity between labeled keypoints and candidate locations, and uses pairwise geodesic distance information as *second order* regularization.

## 7. Implementation Details

The pipeline of our method is visualized in Fig. 10. The features and the pairwise geodesic distances of ground-truth keypoints on the few-shot samples can be computed in advance. We use PyTorch3D [34] for the rendering of shapes, and process the rendered views with the pre-trained vision models. The runtime of the feature extraction thus depends on various factors, like the number of views, the complexity of the rendered mesh, the amount of points for which we extract features and the 2D feature extractor that is used. The feature computation normally takes between 10 to 20 seconds for one shape in our experiments on one NVIDIA A40 GPU.

As the keypoint candidate optimization is non-linear, we use PyTorch with gradient descent to solve the optimization problem. We initialize the matrix $S$ (see Sec. 3.2) with random values from a normal distribution and apply a softmax per row to ensure the right-stochastic character of the matrix $S$ at every optimization step. Using a GPU, the optimization process (5000 steps) can be completed in about 10 seconds. We make our code available under the following URL: https://github.com/wimmerth/back-to-3d-few-shot-keypoints.

### 7.1. Keypoint optimization hyperparameters

To further steer the optimization toward the selection of one clear correspondence per keypoint, instead of possibly averaging over multiple candidates and their features in the optimization, we define an optional selection reward

$$R_{\text{selection}} = \sum_j (\max_i \hat{S}_{ij} - \frac{1}{n} \sum_i \hat{S}_{ij}), \tag{4}$$

and formulate an extended objective function as

$$L = L_{\text{feature}} + \alpha L_{\text{distance}} - \beta R_{\text{selection}} \tag{5}$$

with two weighting parameters $\alpha, \beta$.

- The weighting-parameter $\alpha$ is dependent on the feature dimensionality and magnitude. We find that setting $\alpha = 4$ works well with back-projected DINO features (see Fig. 11), but we also adjust this parameter for the various other features used in our experiments.
- In our experiments, we find that setting $\beta = 0$ in the optimization and thus not taking the selection reward into account works the best (Fig. 11), as the optimization is less sensitive to the random initialization of the selection matrix $S$ without it.
- The choice of $\sigma$ for Gaussian re-weighting of features (Eq. 2) depends on the quality and scale of the given mesh. In the best case, if we work with clean meshes, we do not need to apply re-weighting since all points of the shape will be (recognized as) visible from some viewpoint. However, if this is not the case, as with some ShapeNet meshes, we find that setting $\gamma \in [0.001, 0.005]$ works quite well for shapes normalized to a unit box scale.

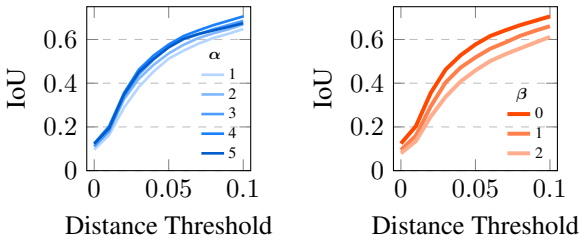


Figure 11. Influence of the weighting terms $\alpha, \beta$ in the optimization objective on the keypoint detection results. The best weights with DINO features were empirically found to be $\alpha = 4$ and $\beta = 0$, thus effectively removing the selection reward from the objective.

### 7.2. Hungarian method baseline

An alternative baseline to nearest-neighbor matching in the feature space is using the Hungarian method to solve the

linear assignment problem with the similarity of candidate features to keypoint features as costs. We show the results using a variant, the Jonker-Volgenant algorithm, in Tab. 1. While this method avoids the collapse observed with simple nearest-neighbor matching in some cases, it is not aware of the spatial relation between the detected keypoints and using our proposed optimization module instead improves over the results by an average of 37%, thus strongly supporting our design choice.

Table 1. Comparison of IoU scores with varying distance thresholds against proposed baseline using Hungarian matching.

| Dist. Threshold | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 | 0.10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B2-3D (ours) | 0.12 | 0.20 | 0.36 | 0.46 | 0.53 | 0.58 | 0.62 | 0.64 | 0.67 | 0.69 | 0.71 |
| Hungarian | 0.08 | 0.13 | 0.23 | 0.31 | 0.37 | 0.42 | 0.47 | 0.50 | 0.53 | 0.55 | 0.58 |

## 8. Results on real-world scans

In addition to experiments on the KeypointNet dataset, we qualitatively evaluate our method on real-world scans from the Objaverse dataset [15]. As there are no ground-truth keypoint annotations given, we manually annotate keypoints on a few cars and apply B2-3D to a few unlabeled cars. We observe that, contrary to the experiments on the KeypointNet dataset, using texture information of the models slightly improves the results which can be explained with the higher quality of the texture compared to ShapeNet meshes.

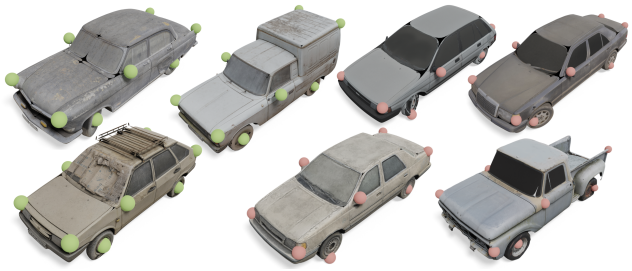

Figure 12. Few-shot keypoint detection (red) on real-world scans, given manually annotated source shapes (green).

## 9. Additional feature analysis and properties

### 9.1. Uptake of local geometry changes

We investigate the reaction of the back-projected features to small modifications of the shape. We expect them to slightly change for the affected regions while remaining similar for non-affected regions. The visual results of this experiment can be seen in Fig. 13.

If we slightly stretch the rear part of the aircraft body, we notice that the features of the aircraft change slightly from the point where we stretch it to the tail. This is interesting to observe because only the fuselage was changed, while the stabilizers in the rear remained untouched. We suspect that the change in the relative sizes of the different parts influences these small changes in the features.

Moving the landing gear on the top of the aircraft changes not only the features of the landing gear, but also those of the part of the aircraft to which it is attached. While it could be argued that the coarse patch size of the extracted features affects this change by ”leaking” onto the area behind the landing gear, features for other regions that are close to the landing gear did not change drastically. This leads us to conclude that the features back-projected to the modified shape also capture the change in semantics for the affected part of the aircraft.

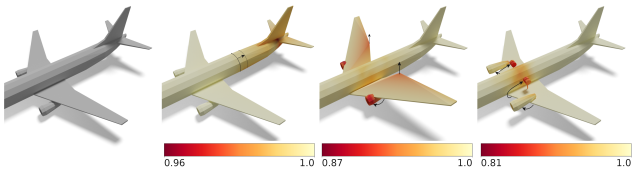


Figure 13. Change of the computed features (measured in cosine similarity) when applying small modifications (indicated with black arrows) to the original shape (left). The back-projected DINO features seem to generally react well by changing in the affected areas while staying the same in unaffected areas.

### 9.2. Semantic stability with varying shapes

In our experiments with axiomatic shape descriptors, we hypothesize that pure geometry-based descriptors are not consistent and informative enough to provide similar features for similar points on shapes of the same object class that have a different geometry. To test this conjecture, we conduct another small qualitative experiment. We extract features for each shape and apply a PCA to all points together for visualization, as described in Section 4.2. In order to have a balanced PCA computation between the meshes with different numbers of vertices, we sample 2048 points from each shape’s surface which we use to fit the PCA. For visualization, we then apply the fitted PCA to the feature computed for each mesh vertex to obtain the colored shapes shown in Fig. 14.

We would want shapes to be colored approximately the same, as similar parts over different shapes should also have similar features. The results (Fig. 14) show that this is far more the case for the back-projected DINO features when compared to the HKS features. It is remarkable that the features are consistent even for shapes that are fairly different, such as different airplane types.

Table 2. Back-projected features are a strong backbone for part segmentation transfer. Label transfer results measured with average IoU.

| | | *pla.* | *bag* | *cap* | *cha.* | *ear.* | *gui.* | *kni.* | *lam.* | *lap.* | *bik.* | *mug* | *pis.* | *roc.* | *ska.* | *tab.* | *avg.* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [24] | IDC | 60.1 | 56.2 | 59.7 | 72.2 | 45.3 | 81.5 | 66.4 | 42.6 | 88.5 | 40.5 | 87.5 | 66.4 | 37.2 | 50.7 | 70.4 | 61.7 |
| [13] | CPAE | 61.3 | 59.3 | 61.6 | 72.6 | 55.5 | 78.9 | 71.3 | 53.2 | 89.9 | 55.4 | 86.5 | 66.2 | 40.2 | 61.8 | 72.5 | 65.8 |
| [3] | NCP | 63.7 | 66.7 | 68.7 | **80.2** | 59 | 78.8 | 72.5 | **61.9** | **91.4** | 57.2 | 89.5 | 61.4 | 44.2 | 63.6 | **79.2** | 69.2 |
| Ours | DINO | **64.8** | **75.1** | 67.5 | 72.1 | **77.3** | **83.6** | 71.7 | 57.9 | 89.6 | **64.3** | **91.0** | **73.2** | **47.9** | **65.8** | 63.1 | **71.0** |
| | CLIP | 59.4 | 66.9 | **73.4** | 62.9 | 70.9 | 76.4 | 68.1 | 52.8 | 84.1 | 57.3 | 86.4 | 66.1 | 43.4 | 63.6 | 62.2 | 66.3 |
| | EffNet | 56.6 | 69.5 | 63.2 | 64.4 | 72.1 | 81.6 | 70.8 | 50.6 | 88.5 | 59.5 | 82.6 | 63.5 | 44.9 | 60.2 | 64.2 | 66.1 |
| | SAM | 36.8 | 65.3 | 57.1 | 59.7 | 56.6 | 79.0 | **72.8** | 51.8 | 75.8 | 43.6 | 49.0 | 56.6 | 32.4 | 46.8 | 53.8 | 55.8 |

(a) HKS features on different shapes of the same class.

(b) Back-projected DINO features on different shapes of the same class.

Figure 14. The back-projected features produce more consistent and distinctive features for similar points on different shapes.

## 10. Part Segmentation Transfer

To further validate the strength and generalizability of the back-projected features, we evaluate them on the task of part label transfer. In our experiments, we follow the setup of Cheng et al. [13], where the goal is to transfer part segmentation labels from one shape to another using the labels from the ShapeNet part dataset [44].

In our experiments, we back-project the features onto the 3D shape, as described in Section 3.1 of the main paper, and perform a $k$-nearest neighbor classification in the feature space, querying the points on the new shape and retrieving the best matching label for each point.

Using DINO features, we outperform the previous state-of-the-art methods in 9 of the 15 categories, with an increase of the average IoU over all classes by almost three percent (see Tab. 2). The semantic information enables gains of up to 18.3% IoU over previous methods for selected classes. While these results underline the strong performance of back-projected features, we want to emphasize the simplicity of the used approach: The nearest-neighbor-based classification in feature space is not informed by the spatial connectivity of points, etc., and the used approach is much faster than the previously proposed methods that require additional optimization [3, 24].

Our experiments on this part segmentation transfer further highlight the superiority of DINO features, as the back-projected DINO features outperform other back-projected features on this task. As with keypoint detection, CLIP and EfficientNet features provide fairly similar results. Even though the SAM model was trained as a foundation model for segmentation tasks, the extracted features seem to not be able to provide the necessary distinctiveness between different object parts. We suggest investigating the reasons for this behavior further, but it could possibly be explained by the absence of SAM's powerful decoder that further processes the extracted features and is possibly responsible for the performance increase in the 2D setting.