Clustering Visual Similar Objects for Enhanced Synthetic Image Data for Object Detection

Julian Rolf *, Detlef Gerhard, Pero Kosic

Posted Date: 26 August 2024

doi: 10.20944/preprints202408.1793.v1

Keywords: similarity analysis; object detection; deep embedded clustering

 doi:10.20944/preprints202408.1793.v1

Article

Clustering Visual Similar Objects for Enhanced Synthetic Image Data for Object Detection

Julian Rolf *, Detlef Gerhard and Pero Kosic

Digital Engineering Chair, Ruhr University Bochum, Universitätsstraße 150 44780 Bochum, Germany
* Correspondence: julian.rolf@rub.de

Abstract: Object detection often struggles with accurately identifying visually similar parts, a challenge commonly faced in industrial applications. To address this issue, we propose a clustering methodology based on the visual similarity of 3D object models. This approach is particularly effective when integrated with synthetic image generation, as both processes rely on 3D models. In a case study, we observed a 22 % increase in classification performance on a validation dataset when training an object detector on visually similar groups rather than on all classes, suggesting the potential of our method as a baseline for a multi-stage classification scheme.

Keywords: similarity analysis; object detection; deep embedded clustering

1. Introduction

In recent years, the performance of machine learning algorithms used for object detection has gained a significant incline. Therefore, potential industrial use cases for this technology have been identified and tested. Even though the quality of these models has improved steadily, they still require a large amount of training data before they can be used in the field. Although some zero-shot object detection models have recently been published, which are designed to work without further training, they usually cannot be used for object identification in an industrial environment without adaptation due to the uniqueness of the objects. Because collecting and annotating image data for object detection involves a great amount of manual effort, concepts for using synthetic data have been investigated. Synthetic data for object detection can be generated by rendering 3D-CAD models in a virtual environment. This kind of data collection can be fully automated and is also error free but it suffers from the *Sim2Real* gap, which describes the difference between real and synthetic data. Furthermore, groups of industrial parts in a particular application are often visually similar which makes a distinction quite difficult. When dealing with these two problems at the same time, the use of synthetic data can be unfeasible. In this paper, we propose a concept to measure the visual similarity of 3D models to identify subsets of similar parts using the objects 3D models. Thus, the concept can be directly combined with a synthetic image generator and is also generally applicable if the objects 3D models are available. The concept will be validated, and potential benefits will be discussed. The paper starts with related work on synthetic data used for object detection and similarity analysis for 3D objects. Afterwards, the concept for clustering visual similar objects based on their 3D models will be presented. The Outcomes section describes the effectiveness of the proposed approach by comparing the validation classification loss of multiple object detectors. Finally, a conclusion is given and potential future research aspects are outlined.

1.1. Related Work

1.2. Synthetic Data for Object Detection

In contrast to traditional image processing methods, modern methods based on machine learning (ML) do not require an algorithmic description of the procedure, but they do require a large amount of data to carry out a learning process. The collection, annotation and labeling of the data required

 doi:10.20944/preprints202408.1793.v1

for training a ML model for object detection is problematic and time-consuming. Depending on the complexity of the use case, several thousand images of the objects are required [1]. This process is associated with a high manual workload, is prone to errors. Therefore, the use of synthetic data offers a promising alternative to the manual collection of ML training data. This type of data can be generated automatically by rendering from the corresponding 3D models and automatically labeled without errors [2,3]. The use of specifically tailored synthetic data for a certain use case to train object detectors has already been applied successfully [3,4]. To simplify the generation of synthetic data Greff et al. published Kubric, which is a framework specialized for synthetic image data generation [5]. Automated access to image databases for backgrounds and interfaces for importing 3D objects allow a large amount of synthetic data to be generated in a short time. Frameworks of this type also offer the potential to be connected to existing and established software systems for product lifecycle management (PLM) in order to create consistent, automated processes [6].

1.3. Object Similarity Analysis

Concepts of similarity analysis for 3D models are based on encoding the 3D model into a comparable representation, often referred to as a signature, descriptor or embedding. Obuchi et al. apply two-dimensional Fourier transforms to rendered depth images of an object from multiple perspectives to create descriptors for each of the selected perspectives [7]. These can then be analyzed using metrics such as cosine distance or similar evaluations. While this approach only encodes visual properties of the 3D models, other approaches focuses on encoding volumetric features [8] or using rule based models in order to achieve accurate representations [9].

In recent years, contrastive models have gained much attention for creating meaningful representations for various kinds of inputs. These models try to learn similarities or dissimilarities on the training data. Known contrastive models are CLIP [10] or DINOv2 [11]. These models differ in their training routines, but both result in a meaningful embedding space for the model. Due to the vast amount of data, these models can be used in a zero-shot manner, meaning additional fine-tuning is not mandatory. Nguyen et al. use the DINOv2 model to generate descriptors based on RGB renderings of 3D models [12]. The authors use the cosine distance to classify image patches of real objects, by comparing an embedding of the image patch and the 3D models. Xiang et al. use the CLIP model for remote sensing scene classification, achieving approximately 85% to 95% accuracy depending on the dataset [13].

Using the descriptors of machine learning models as an input for clustering algorithms for object classification is proposed by Xie et al. as deep embedded clustering, which can lead to impressive results without any prior knowledge of the object classes [14]. Nevertheless, fine-tuning a contrastive model like CLIP for a specified use case can still result in a boost in performance and should be considered if an adequate amount of training data is available [15].

2. Materials and Methods

In this section, the concept for clustering visual similar objects is presented. An overview of the concept is depicted in Figure 1. The algorithms used for the proposed process are presented and alternatives are discussed to allow future research to be carried out. The concept can be divided into three stages, a pre processing stage, the main clustering procedure and a post processing stage. Furthermore, a procedure for fine-tuning the involved contrastive machine learning model for the intended use case is presented.

 doi:10.20944/preprints202408.1793.v1

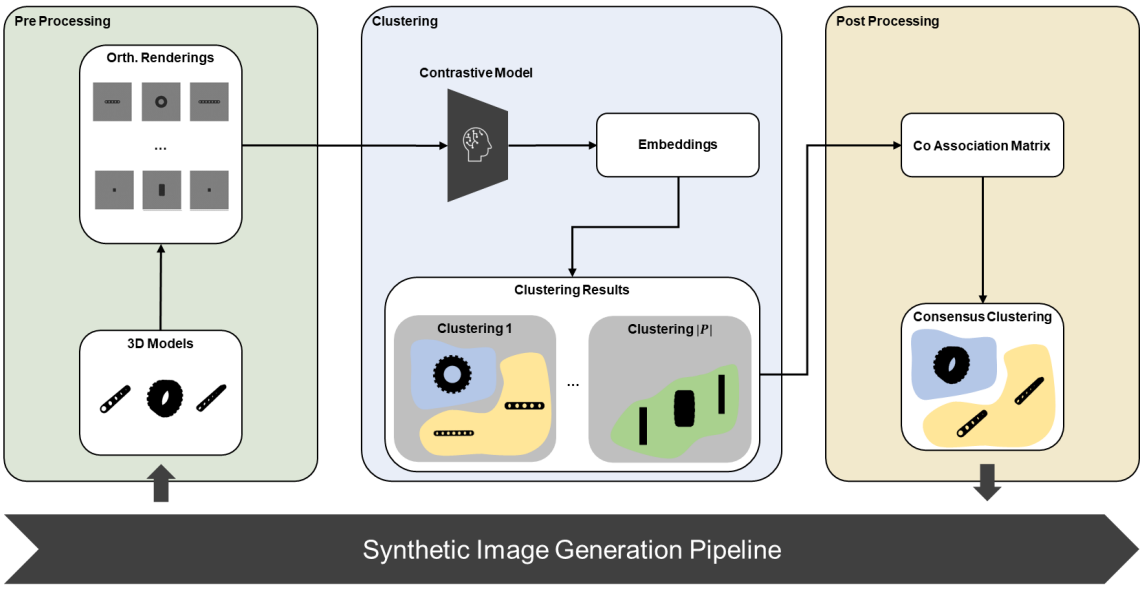

Figure 1. Overview of the proposed concept separated into 3 stages with the 3D models as the input and the final consensus clustering as the result

2.1. Pre Processing

In the first stage, the 3D models are rendered from different perspectives P (e.g. front, back, upper, lower, left and right). Depending on the visual complexity of the objects, the amount of perspectives can be adjusted respectively. The objects are uniformly scaled so that the object's maximum dimension equals 1 unit in the virtual environment, to guarantee a complete capturing of the objects. The objects are placed at the origin of the environment. To avoid shadows being cast, lights are placed at each of the selected perspectives facing the object. The cameras are placed with a distance of 1 unit away from the objects. The renderings are made from an orthogonal projection. Each rendering has a resolution of 224x224 pixel, matching the input size of the latter used DINOv2 model. Each image includes three color channels and an alpha channel. The alpha values of the background are set to 0 and its corresponding RGB values are later on replaced by a white & black checkerboard pattern. Doing so guarantees for a sharp contrast between the object and the background, even if some of the objects have a dark or white surface color. The rendering procedure is carried out for every perspective and every object in the set, resulting in $|P| * N$ images, with N being the total number of 3D models.

2.2. Clustering

Afterwards, each image is fed into the contrastive machine learning model DINOv2. This results in a high dimensional numerical vector representation (embedding) for each image. These Vectors will be used to compare the similarities of the captured objects using a clustering algorithm. In theory, clustering algorithms can be directly applied to the high dimensional vectors. However, reducing the dimension of the data points can help to reveal the intrinsic structure of the data, leading to a better separation of clusters [16], while also reducing noise of the higher dimensions [17,18]. Furthermore, it can also improve the performance of the clustering algorithm, depending on the algorithm being used. Additionally, lower dimensional data points can easily be visualized. Therefore, the t-distributed stochastic neighbor embedding (t-SNE) algorithm is applied to reduce the dimension of the data points to two dimensions. Alternatively, the use of the uniform manifold approximation and projection technique (UMAP) can be considered.

With the dimension of the data points reduced to two dimensions, the clustering algorithm mean shift is applied. This density based clustering algorithm has the advantage of being almost non-parametric. It only relies on the bandwidth parameter, which can be estimated using the data

 doi:10.20944/preprints202408.1793.v1

points to be clustered. Another required property of this clustering algorithm is that it does not require any prior knowledge of the number of existing clusters, like k-means, as the goal is to identify similar objects by clustering their descriptors. An alternative clustering algorithm with similar properties is the HDBSCAN algorithm. In contrast to mean shift, HDBSCAN provides more flexibility with different parameters that can be configured. The clustering algorithm is applied to each perspective individually, resulting in an ensemble of $|P|$ different clustering results.

2.3. Post Processing

In the post-processing stage, the different clustering results are analyzed to generate a single, consensus clustering. This is done by creating an $N * N$ association matrix, representing how often objects n_i and n_j are in the same cluster, normalized by the total number of different clustering results in the ensemble $|P|$. By exceeding a predefined threshold, the objects are considered similar and are placed in the same final cluster.

2.4. Fine-Tuning

While this concept can be applied without any adjustments to a pre-trained contrastive model, fine-tuning to a specialized use case may result in a better performance [15]. Therefore, this subsection presents a concept of fine-tuning the contrastive model, aiming for a better clustering performance.

As the ML model is used to analyze the similarity of industrial components, a large and comprehensive set of 3D models of industrial components is required for fine-tuning. Following the rendering procedure in section 3.1, the 3D models can be rendered from the predefined perspectives. The triplet loss, proposed by Chechik et al. [19], can be applied to guide the learning process. It consists of three components, an anchor, a positive and a negative input. The anchor can be a randomly select sample of the training dataset. The triplet loss aims at minimizing the distance between the anchor and the positive input, while maximizing the distance of the anchor and the negative input, in the embedded space. For a given anchor, a positive sample should be an image of a similar object, or an image of the same object rendered from a different perspective. A negative sample should in contrast be an image of an object of different category or type of the anchor, to ensure a dissimilarity between the anchor and the negative. It is therefore necessary to group the objects in the dataset based on their similarity. In addition, the data set should include a sufficient number of 3D models so that the zero shot capability of the contrastive model is preserved.

3. Results

This chapter presents the effectiveness of the proposed concept. The concept is applied to a set of LEGO pieces, in order to match visually similar parts. Afterwards, a synthetic image dataset is created and used to train three object detectors. The first detector has to identify each part individually, while the second detector has to identify the groups of similar objects based on the similarity matching. The third detector also has to identify groups of similar objects, which are created using a fine-tuned version of the contrastive model. Finally, the detectors are compared by their classification loss on the validation data.

3.1. Clustering without Fine Tuning

First, a selection of objects to be detected is set up. In total 20 LEGO pieces, with different color and shape, are selected. Some of the objects have strong visual similarities to each other, while others are deliberately different. This guarantees difficulties in the classification task for an object detector. The LEGO parts are rendered from the front, back, upper, lower, right and left perspective. The background is set to a checkerboard pattern in order to maintain the contours of the objects. Afterward, the embeddings obtained from inferencing the contrastive model DINOv2 using the ViTs-14 checkpoint are reduced in dimension to 2D using the TSNE algorithm, and then grouped using the Mean Shift

 doi:10.20944/preprints202408.1793.v1

algorithm. A threshold of 0.5 is set for the consensus clustering. The final clustering is shown in Figure 2.

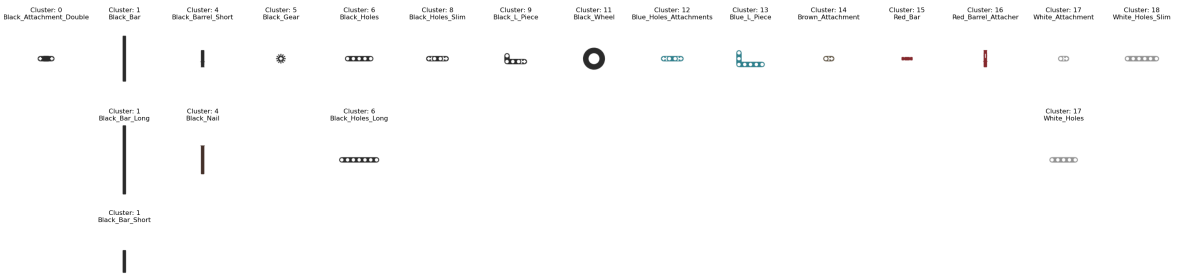

Figure 2. Embedded clustering of the selected LEGO parts

To evaluate the effectiveness of the approach, three object detectors are trained. All three use the same training and validation images, but differ in the class assignment of the objects. While the data set for the first detector has exactly one class for each object, the clustering in Figure 2 is used as the basis for the class assignment of the second data set. In the following subsection, the third detector is trained to identify a clustering based on a fine-tuned version of the contrastive model.

A data generation pipeline is implemented to create the required amount of images synthetically. During the synthetic data generation, 1 to 5 objects are placed inside a virtual environment, with random orientation. The camera is placed at a random point on a half sphere around the origin, within a variable radius. Given the camera's distance to the origin and its FOV, a view plane is calculated. The objects are placed randomly on that plane to ensure visibility to the camera. To differ background and lighting conditions, an HDRI texture is randomly sampled for each image. In total 1000 images are created this way, used entirely for training.

For the validation images, a more realistic data generation pipeline is implemented. The objects are placed randomly above a flat surface. A physics simulation is then carried out to calculate physical accurate orientations of the objects on the surface. This is achieved by simply dropping the objects from the specified height onto the surface. The camera is placed at a variable height over the surface, facing downwards. The minimum and maximum height is set to match the minimum and maximum radius of the half sphere, the camera is placed on during the training data generation. Lighting conditions are again set by sampling a random HDRI texture for each image. 400 images are generated this way, using 200 as validation data and the other 200 as training data. Including 200 samples into the training dataset is because to an otherwise high amount of false positives due to the shadows of the objects on the flat surface. The synthetic images are shown in Figure 3.

Figure 3. Synthetic images used for training and validation

For training, the pre-trained object detection model YOLOv8 in its small version is fine-tuned. The training is stopped when no further improvement on the validation data is achieved for 50 epochs, measured by the mean average precision of the model. Stochastic gradient descent with an initial learning rate of 0.01 is used, in addition to the default augmentation and learning configuration of the

 doi:10.20944/preprints202408.1793.v1

ultralytics python module. Figure 5 shows a plot of the validation classification error during training for the different object detectors. In contrast to an object detection model without any clustering of similar objects, a second detector, trained to identify the resulting clusters, achieves an improvement of approximately 22 %. The results are listed in Table 1.

3.2. Clustering with Fine-Tuning

For fine-tuning the contrastive model, the training procedure described in the concept section is carried out. As for the dataset used to fine tune the contrastive model, the DMU-Net dataset is chosen [20]. It consists of 2000 CAD models, grouped into 34 categories. Rendering each CAD model from six perspectives, results in a total of 12000 training images. For the triplet loss, a positive sample is an image of the same object as the anchor, but from a different perspective. A negative sample is a random image of an object from a different category as the anchor. Additionally, the images are augmented by applying a random rotation and a random horizontal or vertical flip during training. The batch size is set to 16 and the learning rate is set to 1e-4, using the ADAM optimizer. The training is conducted for 10 epochs. Afterwards, the clustering procedure is reapplied, resulting in the clustering illustrated in Figure 4, and a third object detector is trained.

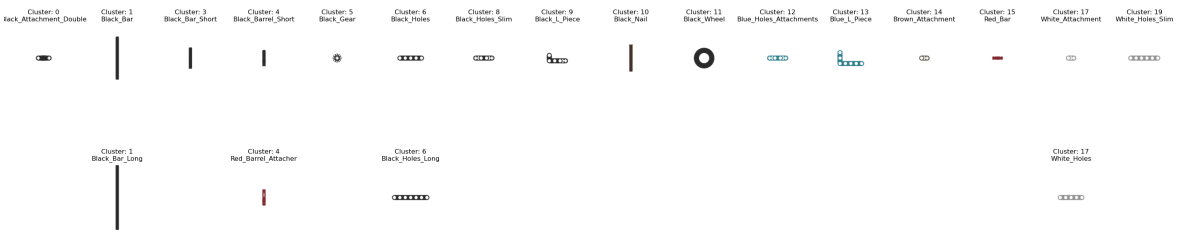

Figure 4. Embedded clustering of the selected LEGO parts using a fine-tuned contrastive model

Figure 5 indicates the validation classification loss of all three object detectors during training, while the numerical results can be found in Table 1. The improvement of the classification error is almost identical to that of the detector trained on the clustering of the standard DINOv2 model and is in a range without statistical significance. However, an additional cluster is identified, which improves the interpretability of the detector results. This result could be achieved by training the contrastive model for only 10 epochs and on just 12000 images.

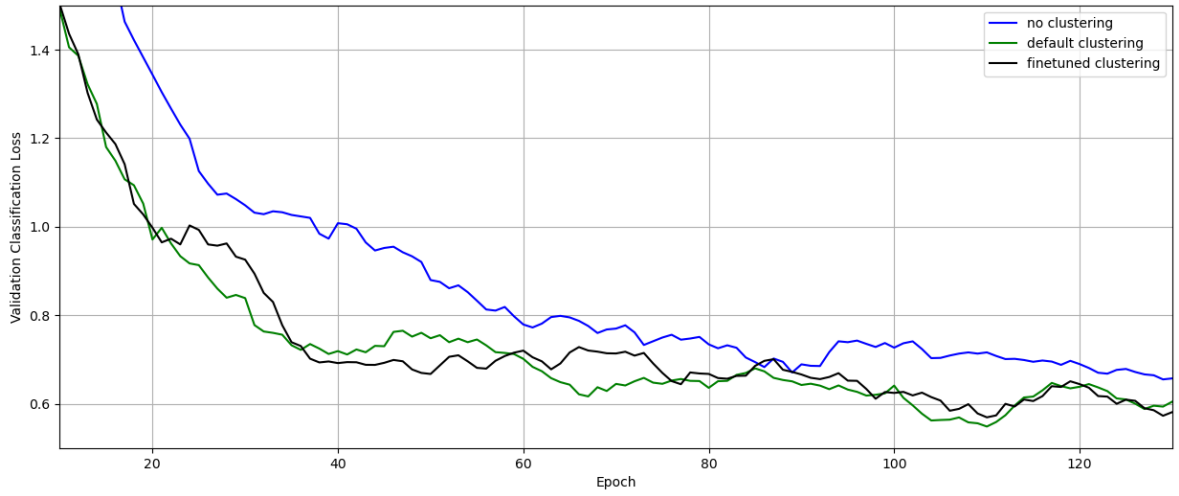

Figure 5. Classification error during training on the validation data

 doi:10.20944/preprints202408.1793.v1

Table 1. Validation classification error

Contrastive Model	Clusters	min. val. cls. loss	improvement
-	20	0.546	-
DINOv2	15	0.426	22.03 %
Finetuned DINOv2	16	0.435	20.44 %

4. Discussion

In this work, we introduced a methodology for grouping visually similar objects using their 3D models for an embedded clustering technique to address challenges in object detection. This approach not only helps assess the applicability of an object detector in specific use cases but also establishes a baseline for implementing a multi-stage classification strategy. Although designed to integrate seamlessly with synthetic data generation pipelines, our method can also be applied in scenarios with real data, if the corresponding 3D models are available.

Our experimental results validate the effectiveness of this approach in the given use case, demonstrating its potential for broader application. For future research, we plan to evaluate the concept with a larger set of objects to confirm its general applicability. Additionally, expanding the dataset and extending the fine-tuning process of the contrastive model are anticipated to further enhance the robustness and accuracy of the proposed method.

Author Contributions: The article is written by Julian Rolf and supervised by Detlef Gerhard. Acquiring the data used for training and implementing the source code was done by Julian Rolf and Pero Kosic

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: We will soon release the git repository including the source code and the data used for training

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Riedel, A.; Gerlach, J.; Dietsch, M.; Herbst, S.; Engelmann, F.; Brehm, N.; Pfeifroth, T. A deep learning-based worker assistance system for error prevention: Case study in a real-world manual assembly. *Advances in Production Engineering & Management* **2021**, *16*, 393–404. https://doi.org/10.14743/apem2021.4.408.
2. Židek, K.; Lazorík, P.; Piteľ, J.; Pavlenko, I.; Hošovský, A. Automated Training of Convolutional Networks by Virtual 3D Models for Parts Recognition in Assembly Process. In *Advances in Manufacturing II*; Trojanowska, J.; Ciszak, O.; Machado, J.M.; Pavlenko, I., Eds.; Springer International Publishing: Cham, 2019; pp. 287–297.
3. Tao, W.; Lai, Z.H.; Leu, M.C.; Yin, Z.; Qin, R. A self-aware and active-guiding training & assistant system for worker-centered intelligent manufacturing. *Manufacturing Letters* **2019**, *21*, 45–49. https://doi.org/10.1016/j.mfglet.2019.08.003.
4. Lai, Z.H.; Tao, W.; Leu, M.C.; Yin, Z. Smart augmented reality instructional system for mechanical assembly towards worker-centered intelligent manufacturing. *Journal of Manufacturing Systems* **2020**, *55*, 69–81. https://doi.org/10.1016/j.jmsy.2020.02.010.
5. Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y.; Duckworth, D.; Fleet, D.J.; Gnanapragasam, D.; Golemo, F.; Herrmann, C.; Kipf, T.; Kundu, A.; Lagun, D.; Laradji, I.; Liu, H.T.D.; Meyer, H.; Miao, Y.; Nowrouzezahrai, D.; Oztireli, C.; Pot, E.; Radwan, N.; Rebain, D.; Sabour, S.; Sajjadi, M.S.M.; Sela, M.; Sitzmann, V.; Stone, A.; Sun, D.; Vora, S.; Wang, Z.; Wu, T.; Yi, K.M.; Zhong, F.; Tagliasacchi, A. Kubric: a scalable dataset generator **2022**.
6. Rolf, J.; Wolf, M.; Gerhard, D. Investigation of an Integrated Synthetic Dataset Generation Workflow for Computer Vision Applications. In *Product Lifecycle Management. Leveraging Digital Twins, Circular Economy,*

and Knowledge Management for Sustainable Innovation; Danjou, C.; Harik, R.; Nyffenegger, F.; Rivest, L.; Bouras, A., Eds.; Springer Nature Switzerland: Cham, 2024; Vol. 702, *IFIP Advances in Information and Communication Technology*, pp. 187–196. https://doi.org/10.1007/978-3-031-62582-4{\textunderscore }17.

7. Ohbuchi, R.; Nakazawa, M.; Takei, T. Retrieving 3D shapes based on their appearance. Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval - MIR '03; Sebe, N.; Lew, M.S.; Djeraba, C., Eds.; ACM Press: New York, New York, USA, 2003; p. 39. https://doi.org/10.1145/973264.973272.
8. Kaku, K.; Okada, Y.; Niijima, K. Similarity measure based on OBBTree for 3D model search. Proceedings. International Conference on Computer Graphics, Imaging and Visualization, 2004. CGIV 2004., 2004, pp. 46–51. https://doi.org/10.1109/CGIV.2004.1323959.
9. Zehtaban, L.; Elazhary, O.; Roller, D. A framework for similarity recognition of CAD models. *Journal of Computational Design and Engineering* **2016**, *3*, 274–285. https://doi.org/10.1016/j.jcde.2016.04.002.
10. Ma, Y.; Xu, G.; Sun, X.; Yan, M.; Zhang, J.; Ji, R. X-CLIP: End-to-End Multi-grained Contrastive Learning for Video-Text Retrieval. Proceedings of the 30th ACM International Conference on Multimedia; Magalhães, J.; Del Bimbo, A.; Satoh, S.; Sebe, N.; Alameda-Pineda, X.; Jin, Q.; Oria, V.; Toni, L., Eds.; ACM: New York, NY, USA, 2022; pp. 638–647. https://doi.org/10.1145/3503161.3547910.
11. Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Assran, M.; Ballas, N.; Galuba, W.; Howes, R.; Huang, P.Y.; Li, S.W.; Misra, I.; Rabbat, M.; Sharma, V.; Synnaeve, G.; Xu, H.; Jegou, H.; Mairal, J.; Labatut, P.; Joulin, A.; Bojanowski, P. DINOv2: Learning Robust Visual Features without Supervision, 2024, [arXiv:cs.CV/2304.07193].
12. Nguyen, V.N.; Groueix, T.; Ponimatkin, G.; Lepetit, V.; Hodan, T. CNOS: A Strong Baseline for CAD-based Novel Object Segmentation, 2023, [arXiv:cs.CV/2307.11067].
13. Li, X.; Wen, C.; Hu, Y.; Zhou, N. RS-CLIP: Zero shot remote sensing scene classification via contrastive vision-language supervision. *International Journal of Applied Earth Observation and Geoinformation* **2023**, *124*, 103497. https://doi.org/https://doi.org/10.1016/j.jag.2023.103497.
14. Xie, J.; Girshick, R.; Farhadi, A. Unsupervised Deep Embedding for Clustering Analysis.
15. Arto.; Dev Vidhani.; Goutham.; Mayank Bhaskar.; Sujit Pal. Fine tuning CLIP with Remote Sensing (Satellite) images and captions, 2021.
16. Becht, E.; McInnes, L.; Healy, J.; Dutertre, C.A.; Kwok, I.W.H.; Ng, L.G.; Ginhoux, F.; Newell, E.W. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology* **2018**. https://doi.org/10.1038/nbt.4314.
17. Laurens van der Maaten.; Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research* **2008**, *9*, 2579–2605.
18. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; Sutskever, I. Learning Transferable Visual Models From Natural Language Supervision.
19. Chechik, G.; Sharma, V.; Shalit, U.; Bengio, S. Large Scale Online Learning of Image Similarity Through Ranking. *J. Mach. Learn. Res.* **2010**, *11*, 1109–1135.
20. Dekhtiar, J.; Durupt, A.; Bricogne, M.; Eynard, B.; Rowson, H.; Kiritsis, D. Deep learning for big data applications in CAD and PLM – Research review, opportunities and case study. *Computers in Industry* **2018**, *100*, 227 – 243. https://doi.org/https://doi.org/10.1016/j.compind.2018.04.005.