

# Bent & Broken Bicycles: Leveraging synthetic data for damaged object re-identification

Luca Piano, Filippo Gabriele Praticò, Alessandro Sebastian Russo, Lorenzo Lanari, Lia Morra, Fabrizio Lamberti

Department of Control and Computer Engineering, Politecnico di Torino, Torino, Italy

{luca.piano, filippogabriele.prattico, alessandrosebastian.russo}@polito.it,  
lorenzo.lanari@studenti.polito.it, {lia.morra, fabrizio.lamberti}@polito.it

## Abstract

*Instance-level object re-identification is a fundamental computer vision task, with applications from image retrieval to intelligent monitoring and fraud detection. In this work, we propose the novel task of damaged object re-identification, which aims at distinguishing changes in visual appearance due to deformations or missing parts from subtle intra-class variations. To explore this task, we leverage the power of computer-generated imagery to create, in a semi-automatic fashion, high-quality synthetic images of the same bike before and after a damage occurs. The resulting dataset, Bent & Broken Bicycles (BB-Bicycles), contains 39,200 images and 2,800 unique bike instances spanning 20 different bike models. As a baseline for this task, we propose TransReID3D, a multi-task, transformer-based deep network unifying damage detection (framed as a multi-label classification task) with object re-identification. The BBBicycles dataset is available at <https://tinyurl.com/37tepf7m>*

**keywords** instance-level retrieval; re-identification; synthetic data; damage detection; transformers

## 1. Introduction

Deep learning has fueled unprecedented advances in tasks such as person re-identification (ReID) [14, 60, 29, 44, 9], vehicle ReID [24, 17] and instance-level object retrieval [64, 3, 7, 54, 49]. The availability of suitable datasets for training and testing ReID systems is a key ingredient to this success. Existing ReID benchmarks, typically focusing on persons [56, 63, 28] and vehicles [30, 31], are limited in size and variety. Even when they include a large number of IDs [30, 56], they generally cover a limited geographical area (e.g., a town or campus circuit) and time window (e.g., a few hours or days). For this reason, the community has recognized the potential of synthetic data for tasks

such as person detection, tracking, and ReID [14, 3]. In addition to the sheer volume of generated data, synthetic generation can increase its variety in terms of background, illumination, weather, pose, etc., so that deep neural networks (DNNs) can incorporate all the invariances needed to generalize in real-world conditions.

In the spirit of pursuing even more robust object ReID, we wish to investigate whether it is possible to make DNNs invariant not only to changes in the environment, but also to changes in the object visual appearance, such as those that could occur due to aging, degradation, damages, or removable/interchangeable parts. Long-term ReID requires the ability to distinguish stable properties over time to account, e.g., for changes in person clothing [47, 19] or seasonal changes in places [32]. Here, we propose the novel task of **damaged object re-identification**, which aims to identify the same object in multiple images even in the presence of breaks, deformations, and missing parts. Besides the theoretical interest, robust object ReID is motivated by practical applications like, e.g., fraud detection and smart contracts in the insurance domain [35].

As a benchmark for this task, we propose to focus on the study of bicycles, which are characterized by challenging intra-class variations and at the same time allow for a wide range of realistic deformations. Unlike landmarks that have unique and distinctive features, bike instances must be separated based on subtle cues (e.g., color, texture, or stick-ers). Deformations are inherently different from occlusions, since object parts are visible but with changes in shape (deformation) or texture (e.g., due to mud, dirt, or rust). Therefore, the insights collected from BBBicycles could be useful for other ReID tasks (e.g., vehicle, person), with similar challenges for long-term ReID. Since acquiring real images of the same bicycle before and after deformation would be prohibitively challenging, we took advantage of computer graphics to generate the Bent & Broken Bicycles (BBBicycles) dataset, which we release as the first dataset for train-

ing and testing DNNs for damaged object ReID.

Our **contributions** can be summarized as follows:

- We design a semi-automatic computer graphics pipeline to simulate different types of damage, breaks, missing parts, and material deterioration. Extensive domain randomization is further employed to train deep networks robust to variations in bicycle pose, background, etc. [51, 50].
- We release the BBBicycles dataset containing 39,000 annotated images. BBBicycles allows DNNs to (learn to) differentiate subtle intra-class variations (including different setups of the same bike model) from deformations occurring due to incidents, or aging.
- We propose TransReID3D (Transformer-based object Re-IDentification & Damage Detection), a novel transformer-based multitask DNN for joint damage detection (DD) and ReID.

## 2. Related work

### 2.1. Transformer-based re-identification

Object ReID is the task of identifying the same object across multiple images, regardless of its pose, illumination, or context. It has many important applications such as intelligent monitoring [24, 59], multi-object tracking and robotics [35, 27], fraud detection [26], etc. The reader is referred to many comprehensive surveys for an introduction to this vast body of literature [64, 24, 7]. In recent years, the Vision Transformer (ViT) architecture [12] has sparked a new wave of transformer-based architectures for many computer vision tasks [23]. Transformer-based ReID solutions can be broadly categorized in *hybrid transformer-CNN* [18, 60, 29] and *pure ViT-based* architectures [17, 49].

Hybrid architectures combine CNNs as a feature extractor with a transformer-based module that tackles the matching and metric learning problem [29, 49, 60, 18]. This approach leverages, on the one hand, CNNs hard inductive biases (e.g., translation equivariance) to work effectively on small- to medium-scale datasets. On the other hand, transformers enable cross-attention mechanisms between pairs of query and gallery images [29, 49]. For instance, the Reranking Transformer [49] concatenates image patches from both the query and gallery images in a single sequence, which is then fed to a final classifier predicting the probability of two images representing the same object.

More recently, a variety of pure transformer-based approaches have achieved state-of-the-art results in several ReID tasks [45, 66, 44, 17]. Compared to CNNs, transformers are better suited to handling long-range dependencies and avoid the use of downsampling operators (e.g., pooling and strided convolutions) that may obscure important visual

details [17]. The available architectures are typically based on a ViT backbone, pre-trained on very large-scale datasets such as ImageNet21K, and modified to extract both local and global features [66, 17, 45].

### 2.2. Synthetic data in deep learning

The use of synthetic data is becoming increasingly popular for training machine and deep learning models. Although it is being experimented in multiple domains like, e.g., bioinformatics [43], natural language processing [54], etc., this approach is indeed expected to bring the largest benefits to the field of computer vision. Synthetic data generation is not only an effective approach to scale data generation and annotation, it can also be used to evaluate the robustness of an algorithm under controlled conditions or to alleviate data privacy issues [61, 9].

A recent survey categorized hundreds of synthetic datasets and the use cases they have been devised for [37]. Initially used to address low-level computer vision tasks such as optical flow [33], synthetic datasets are increasingly used to generate training datasets for high-level tasks such as, e.g., object recognition and detection [38], pose estimation [52], segmentation [34], human action recognition [10] and pedestrian tracking and ReID [55, 14]. Works in this field typically build onto well-known repositories, including millions of virtual models with known categories or properties, which can be programmatically manipulated to automate both data generation and its labelling [5, 25].

Popular approaches for collecting synthetic data also include the use of video games [42, 8, 48, 40, 36], or fusing real and virtual data via compositing techniques and placing, e.g., virtual models onto real background images [13].

One of the main challenges associated with synthetic data is the domain shift between real and synthetic images, which can be tackled through transfer learning or domain adaptation [21, 46, 62]. *Domain randomization* is a technique used to enhance the variability of synthetic data and has been shown to substantially increase performance in the real world [51]. With ever increasing CGI fidelity, the synthetic-to-real domain gap is progressively reducing. Recent exciting results showed that training DNNs on very large and diverse synthetic datasets can outperform using public real datasets on tasks such as pedestrian tracking and ReID, even without fine-tuning on real data [14].

## 3. Dataset

This section describes the semiautomatic CGI pipeline designed to generate the BBBicycles dataset, together with its main properties and distribution.

### 3.1. CGI Pipeline

The CGI pipeline, depicted in Figure 1, consists of two main phases. The first phase is model preparation, which



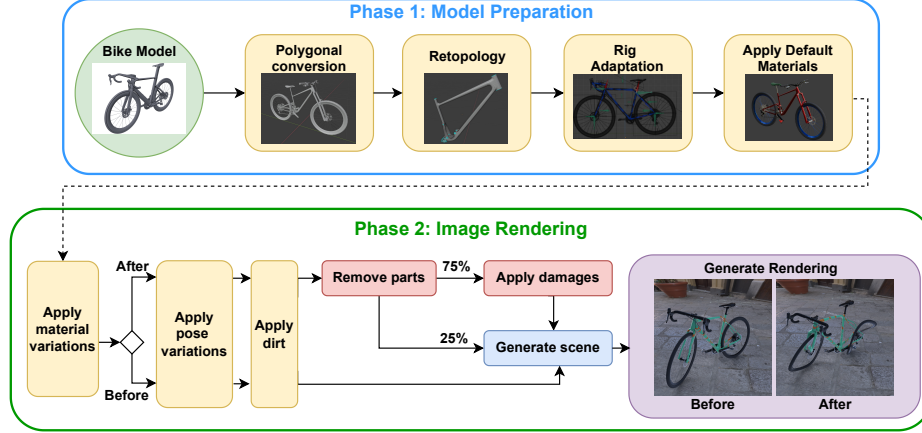


Figure 1: Flowchart illustrating the CGI pipeline. The 3D model is manually prepared (Phase 1) so that it can be easily manipulated by a semi-automatic image rendering script (Phase 2). The script first selects the material textures and colors, thus obtaining a new bike instance (ID). For each ID, multiple images (“before” and “after” the damage) are generated, simulating damages (missing parts, bent and broken frames, etc.) with varying probabilities. Finally, the scene is generated by placing the bike onto a random background.

is mostly manual and performed once for each bike model. In the second phase, a semi-automatic script generates a set of rendered images, depicting multiple views and variations of a given input bike, along with labeling and segmentation information. We sought to create a pipeline that could be applied to generate new datasets with limited human effort and hardware resources. Following this philosophy, we sacrificed some degree of photorealism in favor of reduced rendering time and increased variability. Damages and deformations were implemented based on the classical CGI technique of 3D polygonal meshes armature deformation.

This approach was preferred to, e.g., physics simulations, since it drastically reduces the overall rendering time while maintaining control over the output features desired in terms of missing parts, type of damage, etc. The whole pipeline was implemented in Blender v2.93 [1] and the automatic procedure was scripted in Python as a custom add-on.

**Model preparation.** The input can be either a 3D parametric model (e.g., CAD file) or a polygonal mesh. In the former case, a polygonal conversion is first required to generate a polygonal model. To ensure visually plausible deformations, a retopology operation has to be performed in order to obtain *quad-flow* based topologies with proper vertex density in the parts that will later be subject to deformation. Afterwards, the model is rigged and skinned (i.e., each vertex is associated with a deformation tool of the rig). To make the model easily controlled, we defined a *template rig* that needs to be adapted to the given bike model. The template rig is made up of an “armature”, “lattices”, and “rail

guides” (examples are shown in Appendix A).

More in detail, the template **Armature** includes three groups/layers of “bones”. The red bones are linked to the seat and handlebar meshes (rigid-body movement). The green bones, placed in the salient parts/joints, are used as inverse kinematic controls (targets and poles) by the blue chains. The latter are the so-called deformation bones; only this group was modified by adding/removing bones, if required by the peculiarities of the bike model. These deformation chains are the ones used for the bike frame mesh skinning, whereas other parts (e.g., seat, handlebars, and wheels) are parented (bone relatively) to the dedicated bones of the other two groups. A set of predefined deformations were devised in the form of a pose library to both change the poses of the movable bike components and introduce damages while rendering the images.

The **Lattice** is a three-dimensional non-renderable grid of vertices, a.k.a. deformation cage. Lattices are a convenient way of proportionally deforming a dense mesh with fewer control points since, by deforming the cage, the deformation will be transferred to the associated mesh. The lattices were used to damage the wheels. A set of deformations was devised also in this case in the form of a shape key (a.k.a. blend shape) library. Additionally, **Rail guides** were used to break the bike frame exploiting a boolean mesh operation on a plane that takes the guides as reference.

**Domain randomization and image rendering.** After the 3D model is arranged as described, it is possible to automatically render a variety of different pictures, as described in the following. First, the 3D model is configured by ran-

domly selecting a set of materials (texture, color, and decals) from the material library. A physically based rendering (PBR) material library was defined, from which to pick a suitable material, among several possible choices, for each bike part. *A given combination of 3D model and materials corresponds to a single bicycle instance and is therefore assigned a unique ID.* Second, for each ID, multiple images are generated, “before” or “after” a damage occurs, by applying the following transformations: i) changing the pose of mobile parts (seat, handlebar, pedals and wheels); ii) (optional) applying mud or rust; iii) (optional) damage simulation; iv) point of view selection; and v) background and lighting selection. All deformations are applied randomly with predetermined probabilities and/or ranges. Possible damages include removal of one or more parts of the bike (seat, pedals, handlebar, and wheels), bent frame, broken frame, and wheel deformation.

Finally, the rendered bike must be placed onto a suitable background, adjusting for the specific lighting conditions.

The approach considered in the pipeline takes advantage of the LilyScraper [2], a Blender add-on to use a High Dynamic Range Imaging (HDRI) map as background and light source, in combination with a shadow-catcher plane. The setup of the environment and the lighting was performed once for all models.

### 3.2. BBBicycles characteristics

**Dataset distribution.** The final dataset contains a total of 39,200 images from 2,800 unique IDs (20 models, 140 IDs each). 20 models retrieved from dedicated marketplaces were prepared, including 6 MTBs, 1 Enduro, 6 Road bikes, 1 Circuit, 1 Gravel and 5 Cruiser (following the categorization introduced in [41]). For the textures, we collected five patterns of various styles. Both the base and pattern colors were randomly chosen from a pool of 50 colors. Additionally, 10 different decals containing logos from famous bike brands such as *Bianchi* and *Cannondale* were randomly applied. The background was selected from a pool of 11 different 360° HDRIs, varying bike positioning and illumination by rotating the camera.

For each bike ID, up to 14 renderings were generated, evenly divided in “before” and “after” images as shown in the flowchart (Figure 1). For “before” images, only dirt or rust was applied with 20% probability. For “after” images, dirt/rust was applied with 50% probability, damages to the frame were applied with 75% probability (25% were bent, 25% were broken and 25% were both bent and broken), and finally each removable part (seat, pedals, handlebar, and wheels) was removed (50% probability) or deformed (50% probability). Thus, some of the “after” images are not damaged. Labels for the ReID task were automatically generated based on the bike unique ID assigned by the pipeline.

**Training, validation, and stress test set.** The dataset was split into a training, validation, and test set at the level of bike ID and model to test DNNs’ ability to generalize both across IDs and across models. The validation set includes both models seen and unseen during training, whereas the (stress) test set includes only models that were never seen in either the training or validation set, to ensure that it is sufficiently challenging and representative of real operating conditions. Specifically, the training set contains 25,676 images (1,834 IDs, 14 models), the validation set contains 1,128 images (564 IDs, 12 models), and the stress test contains 840 images (420 IDs, 3 models).

**Real dataset.** A separate dataset of real photos of damaged and undamaged bikes was also collected to test the ability of TransReID3D to generalize to the real domain. We combined a subset of the publicly available DelftBikes dataset [22] with images collected by web scraping from popular search engines and e-commerce sites. The images were manually labeled following the same criteria as those used for the synthetic dataset. A total of 6,292 images were collected, of which 106 presented a Bent (64) or Broken (52) frame. The dataset was split into train, validation and test with a 7:1.5:1.5 split, stratified by damage type.

## 4. Methodology

**Problem setting** We assume that the training set  $D$  consists of  $N$  sequences of synthetic images  $D = \{(x_i^1, \dots, x_i^M)\}_{i=1}^N$ , where all images  $x_i^j$  in a sequence are associated with the same ID  $i$  and represent the same bike instance. We additionally assume that each image is associated with a set of binary attributes, each representing the presence of a specific kind of damage ( $a_i^j \in \mathcal{A} = \{BD, BK, P_n\}$ );  $P_n$  indicates whether the  $n^{th}$  part is present or missing. Given  $D$ , our aim is to learn an embedding space  $x_i^j \in \mathbb{R}^{h \times w \times ch} \mapsto e_i^j \in \mathbb{R}^m$  such that all images associated with a given ID  $i$  are closer in the embedding space than other IDs, regardless of the attributes  $a_i^j$ . We further define the DD task as predicting the values of  $a_i^j$  (multi-label binary classification). At inference time, a query image is compared against the gallery, and the correct ID must be retrieved on the basis of the embedding distance. We assume that the damaged bikes are the queries, inspired by applications in the insurance domain (fraud detection).

**TransReID3D architecture** The TransReID3D architecture for joint DD and ReID, shown in Figure 2, builds on the TransReID [17] architecture, which achieved state-of-the-art performance among ViT-based models for vehicle ReID, and enriches it with an additional multi-label DD branch.

The TransReID architecture [17] builds on the ViT architecture [12], but includes additional components to cap-

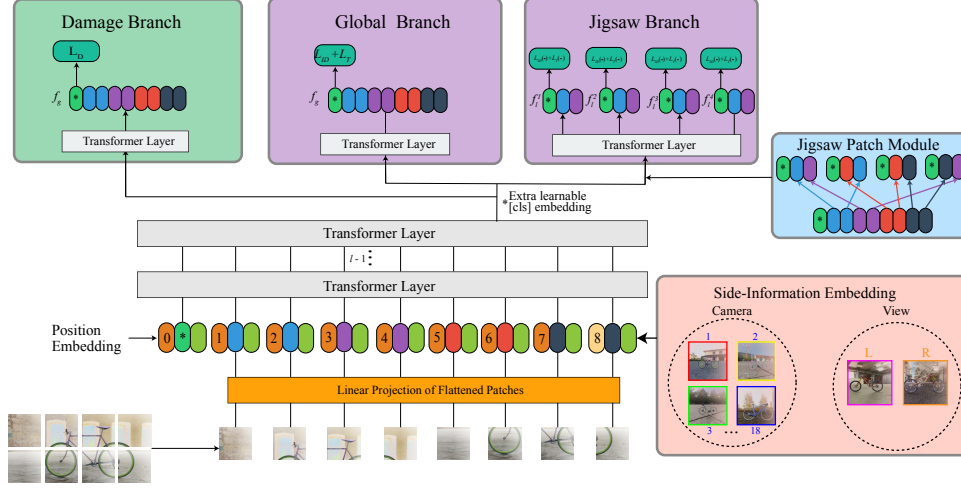


Figure 2: TransReI3D architecture. Embeddings are enriched with position and camera information (side information embedding). A learnable  $[cls]$  token is prepended to the embeddings which are input to a shared backbone. Task-specific branches (DD branch, Global ReID branch and Jigsaw Branch with JPM) include a separate transformer layer to adapt global features to each task. The Jigsaw Module,  $\mathcal{L}_{ID}$ , and  $\mathcal{L}_T$  are described in [17].

ture more robust and fine-grained features. Specifically, the Side Information Embedding (SIE) module encodes non-visual information such as camera or viewpoint, and is input to a transformer encoder together with learnable patch and position embeddings. The global ReID branch and the Jigsaw branch then jointly learn the ReID task, encoding global ( $f_g$ ) and local ( $f_l$ ) features, respectively. The Jigsaw branch is based on the Jigsaw Patch Module (JPM), which shuffles all patches and regroups them into several groups, all of which are input to a shared transformer layer to learn local features  $f_l$ , as detailed in [17].

**Damage branch and multi-task learning.** Multi-task learning is implemented using one shared transformer backbone and an additional separate transformer layer for each task [39, 57]. The DD branch is a multi-label classifier with seven output heads: two for Bent and Broken frame labels, and five for missing parts (front wheel, rear wheel, seat, handlebar or pedals). Each output head takes as input the  $[cls]$  token and passes it through a batch normalization (BN) layer followed by a fully connected (FC) layer.

TransReI3D combines two tasks, one executed on image pairs (ReID) and one executed on individual images (DD). In addition, the ReID task is not defined for real images. For this reason, a multi-task diversion mechanism was implemented which selects the tasks that need to be executed upon the extracted features of each training batch. Hence, synthetic images are forwarded to all branches, whereas real images are directed to the DD branch only.

**Loss computation.** The loss combines the ReID loss, including global and local features, with the DD loss:

$$\mathcal{L} = \alpha \mathcal{L}_{ID}(f_g) + \beta \mathcal{L}_T(f_g) + \gamma \mathcal{L}_D(f_g^a, f_g^p, f_g^n) + \frac{1}{k} \sum_{j=1}^k \left( \mathcal{L}_{ID}(f_l^j) + \mathcal{L}_T(f_l^j) \right) \quad (1)$$

where  $\mathcal{L}_T$  and  $\mathcal{L}_{ID}$  are the triplet loss and the ID cross-entropy loss (which treats each ID as a separate class, as defined in [17]),

$\mathcal{L}_D$  is the DD loss, and  $k$  ( $= 4$ ) is the number of classification heads of the JPM branch. All loss components are calculated on the  $[cls]$  token ( $f_g$ : global branch,  $f_l$ : Jigsaw branch). To compute  $\mathcal{L}_T$ , triplets are online sampled from each batch with hard negative and positive mining.  $\mathcal{L}_D$  is a weighted binary cross-entropy loss:

$$\mathcal{L}_D = \lambda \mathcal{L}_{BD}(\cdot) + \mu \mathcal{L}_{BK}(\cdot) + \nu \frac{1}{n} \sum_{j=1}^n (\mathcal{L}_{P_n}(\cdot)) \quad (2)$$

where  $\mathcal{L}_{BD}$  and  $\mathcal{L}_{BK}$  refer to the Bent and Broken frame labels losses, and  $\mathcal{L}_{P_n}$  to the  $n = 5$  specific missing parts losses. In the case of real images, for the sake of simplicity we consider only  $\mathcal{L}_{BD}$  and  $\mathcal{L}_{BK}$ .

**Domain adaptation.** In the baseline, TransReI3D is trained on BBBicycles and tested on the real data set, without adaptation or fine-tuning. We further explored different

domain adaptation strategies. For *supervised domain adaptation*, we simply leveraged the multi-task training strategy to train the model on real and synthetic data. For *unsupervised domain adaptation*, we experimented with the well-known domain adversarial technique DANN [15] and with partial domain adaptation PADA [4]. Experiments with PADA were motivated by the observation that BBBicycles includes a wider range of bike models and setups compared to the real dataset, and therefore forcing the feature distributions to align could lead to negative transfer. PADA assumes that the target domain contains different labels than the source, whereas in our setting DD labels are the same (additionally, all labels are binary given the multi-label setting). Therefore, we introduced the auxiliary task of bike model classification (model information is available for synthetic images); PADA exploits these predictions to enhance the contribution of (samples of) bike models that are present both in the synthetic and real datasets. Further details are available in Appendix B.

## 5. Experimental settings

### TransReI3D Training and hyper-parameter settings.

All images were resized to  $256 \times 256$ , and normalized with the mean and standard deviation calculated on the synthetic training set. Data augmentation was performed with random color- and texture-preserving transformations (horizontal flip, crop, blurring, and gaussian noise). Each image was split into overlapping  $16 \times 16$  patches, with patch stride set to  $12 \times 12$ . Batches containing either real or synthetic images were alternated, and the real dataset was iterated twice per epoch to counterbalance the smaller size.

For all experiments, the model backbone was pre-trained on ImageNet [11], and the remaining weights were initialized by Kaiming normal initialization [16]. All models were trained for 20 epochs. The SGD optimizer was used with batch size set to 32, momentum to 0.9 and weight decay to  $1e-4$ . The cosine learning rate scheduler was used (initial learning rate 0.01, linear warmup for 5 epochs).

Regarding the loss, we set  $\alpha$ ,  $\beta$  and  $\gamma$  to 1, whereas for  $\mathcal{L}_D$ , we set  $\lambda$ ,  $\mu$ ,  $\nu$  to 0.25, 0.25 and 0.5, respectively.

**Other baselines.** TransReI3D was compared against the Reranking Transformers (RRT) Global retrieval baseline [49]. RRT was trained on BBBicycles for 50 epochs. The training setting is the same as the default one used in the original code, with learning rate of  $1e-3$ , SGD optimizer with 0.9 momentum, batch size 128, weight decay of  $4e-4$ , MultiStep learning rate scheduler with a 0.1 decay at epochs 30 and 40, contrastive loss and ResNet-50 backbone. However, since RRT does not perform damage detection, it was evaluated only on the ReID task.

**Evaluation protocol.** Performance on the ReID task was measured using common metrics for vehicle and object ReID, i.e., mean Average Precision (mAP) and Cumulative Matching Characteristics (CMC) [17]. CMC- $K$ , with  $K = \{1, 5, 10\}$ , represents the average probability of observing the correct identity within the top- $K$  ranked results. Since the gallery contains one instance per bike ID, it is equivalent to Recall@ $K$ . For each pair of images in the validation and stress test, we set the “after” image as Query and the “before” image as Gallery. All images from other IDs (including those derived from the same 3D bike model) were used as distractors.

For the DD task, performance was measured using the Area under the Receiver Operating Characteristic Curve (AUROC), macro-averaged across all labels. For the sake of conciseness, we report only results for Bent and Broken labels, since damages to the frame are more challenging to detect than missing parts. All performance metrics were averaged over three runs.

## 6. Results

### What is the DD and ReID performance of the baseline, with and without real labeled images at training time?

The baseline was trained in two different settings: one assuming that only synthetic data is available at training time (BL), and one assuming that a small sample of labeled images is available at training time (BL+Real). As shown in Table 1, *on the DD task* BL achieves an average AUC of  $92.1 \pm 0.5$  for synthetic images and of  $93.4 \pm 1.5$  for real images. However, we postulate that there is still a domain shift between the synthetic and the real data, since performance on the DD task improved when the network was exposed to the real domain during training (AUC= $97.3 \pm 2.2$ ).

Delving deeper in the DD task, *the performance varies for different damage types on the synthetic dataset*, with higher AUC on Broken ( $100 \pm 0.0$ ) than Bent frames ( $81.5 \pm 2$ ). Bent frames are more challenging to detect since some frames (e.g., Cruiser) may include both straight and curved lines, and BBBicycles includes a range of both subtle and heavy damages. On the other hand, the visual features associated with broken frames are well defined and stable between different bike models.

*On the ReID task*, TransReI3D achieved a mAP of  $85.3 \pm 0.2$  (BL and BL+Real) and a CMC-1 of  $79.8 \pm 0.5$  (BL) and  $79.4 \pm 0.1$  (BL+Real), with minor variations when exposed to real data during training. Figure 3 shows how TransReI3D is able to predict the correct ID and distinguish damage-induced variations from different setups of the same (or similar) bike models.

We further investigated the *effect of the background* on the ReID and DD performance. Specifically, we compared three choices of background: (i) HDRI images, as detailed in Section 3; (ii) random selection from Places365

	Validation					
	Damage Detection		Re-identification (Synthetic)			
	Real AUC	Synthetic AUC	mAP	CMC-1	CMC-5	CMC-10
BL	93.4 ± 1.5	92.1 ± 0.5	<b>85.3 ± 0.2</b>	<b>79.8 ± 0.5</b>	91.9 ± 1.1	96.3 ± 0.5
BL + Real <sup>†</sup>	<b>97.3 ± 2.2</b>	91.4 ± 0.2	<b>85.3 ± 0.2</b>	79.4 ± 0.1	<b>92.9 ± 0.4</b>	<b>96.6 ± 0.4</b>
RRT (Global)	-	-	80.5 ± 1	74.1 ± 1.6	88.3 ± 1.1	93.4 ± 1.2
BG Places365 + Real <sup>†</sup>	96.3 ± 1.9	90.4 ± 0.2	85 ± 0.1	79.0 ± 0.4	92.8 ± 0.3	96.3 ± 0.2
BG Uniform + Real <sup>†</sup>	95.2 ± 3.4	87.4 ± 1.5	48.5 ± 3.4	39.2 ± 1.9	59.4 ± 5.6	66.0 ± 5.7
ReID (single task) <sup>†</sup>	-	-	83.3 ± 1.2	77.0 ± 1.2	91.2 ± 1.5	95.1 ± 1.4
Damage detection (single task) <sup>†</sup>	<b>97.5 ± 1.5</b>	<b>94.5 ± 0.5</b>	-	-	-	-
BL + DANN <sup>‡</sup>	93.9 ± 1.1	91.7 ± 0.9	85.2 ± 0.2	79.4 ± 0.4	92.3 ± 1.0	96.4 ± 0.5
BL + Real + DANN <sup>†</sup>	97.0 ± 1.8	91.0 ± 0.6	85.2 ± 0.5	78.9 ± 0.8	92.8 ± 0.4	96.4 ± 0.7
BL + PADA <sup>‡</sup>	94.4 ± 0.5	90.8 ± 1.2	84.8 ± 0.2	78.6 ± 0.4	92.6 ± 0.3	96.4 ± 0.5
BL + Real + Model labels <sup>†</sup>	96.9 ± 1.9	90.7 ± 1.0	84.6 ± 0.4	77.9 ± 0.7	<b>93.0 ± 0.4</b>	96.6 ± 0.1
BL + Real + PADA <sup>‡</sup>	96.2 ± 3.1	90.9 ± 1.9	84.7 ± 0.1	78.4 ± 0.2	92.4 ± 0.3	<b>96.9 ± 0.6</b>
Stress test						
Baseline	-	94.1 ± 0.2	<b>79.3 ± 0.2</b>	<b>72.5 ± 0.2</b>	87.4 ± 0.3	<b>92.2 ± 0.1</b>
BL + Real <sup>†</sup>	-	93.5 ± 0.23	79.2 ± 0.1	72.1 ± 0.4	88.0 ± 0.1	<b>92.2 ± 0.1</b>
RRT (Global)	-	-	76.1 ± 1.3	65.7 ± 2.3	85.4 ± 2.2	90.6 ± 0.9
BL + DANN <sup>‡</sup>	-	93.4 ± 1.1	78.7 ± 0.5	71.6 ± 0.5	87.9 ± 0.5	91.3 ± 0.7
BL + Real + DANN <sup>†</sup>	-	93.5 ± 0.3	79.1 ± 0.2	71.7 ± 0.2	87.9 ± 0.2	92.1 ± 0.2
BL + PADA <sup>‡</sup>	-	<b>94.2 ± 0.4</b>	79.2 ± 0.4	72.3 ± 0.8	<b>88.1 ± 0.1</b>	<b>92.2 ± 0.5</b>
BL + Real + PADA <sup>†</sup>	-	92.9 ± 1	78.9 ± 0.7	71.9 ± 0.7	87.8 ± 0.8	91.9 ± 0.1

Table 1: Performance on the validation and stress set. All networks trained on synthetic data except for <sup>†</sup> (labeled real images available at training time) and <sup>‡</sup> (unlabelled real images available at training time). Best results are in bold.

[65], and (iii) a simple uniform background (see Appendix C for examples). On the DD task, all transfer scenarios (HDRI (BL) → Real, BG Places365 → Real and BG Uniform → Real) achieved similar results (Table 1). HDRI slightly outperforms Places365: the latter contains a wider range of scenes, but the resulting blend is not as realistic as the proposed HDRI technique. On the ReID task, performance substantially drops when training on a uniform background, as the network does not learn to separate the bike from the background.

**Is multi-tasking beneficial for damaged object re-identification?** We compared TransReID3D against single-task ReID and DD networks – the former reduces to the original TransReID architecture, whereas the latter becomes a ViT-based multi-label classifier. As shown in Table 1, TransReID3D outperforms the single-task ReID architecture both in terms of mAP (85.3 ± 0.2 vs. 83.3 ± 1.2) and CMC (CMC-1 79.9 ± 0.4 vs. 77.0 ± 1.2). This is further confirmed by the performance of RRT (mAP 80.5 ± 1 vs. 85.3 ± 0.2). On the other hand, DD improves in the single-task setting on both real (97.5 ± 1.5) and synthetic (94.5 ± 0.5) images. A possible explanation is that the ReID task forces the network to take into account the entire bicycle, whereas

for DD simpler, more localized visual cues are sufficient. Conversely, the ReID task can leverage the DD labels to learn visual properties invariant to the presence of damage.

**Are feature-level domain adaptation strategies helpful to reduce the synthetic-to-real gap?** The BL results indicate that, at least for the DD task, a certain domain shift still exists. Besides low-level differences due to CGI, we postulate that this domain shift may be attributed to different reasons: on the one hand, few examples of damaged bikes are available; on the other hand, the synthetic dataset contains more bike models (for instance, most images in the Delft Bikes dataset are minor variations of a typical city bike). As detailed in Section 5, we have tested two techniques, DANN and PADA, focusing on the DD task.

When labeled real images are available during training, neither DANN (97.0 ± 1.8) nor PADA (96.2 ± 3.1) outperforms BL + Real (97.3 ± 2.2). On the other hand, if we assume that labels are not available at training time, both DANN (93.8 ± 1.1) and PADA (94.4 ± 0.5) improved over BL (93.4 ± 1.5), but did not match the supervised setting (97.3 ± 2.2). On the ReID task, domain adaptation slightly hurts the performance in terms of CMC-1, bearing however in mind that this task is evaluated only on synthetic images.

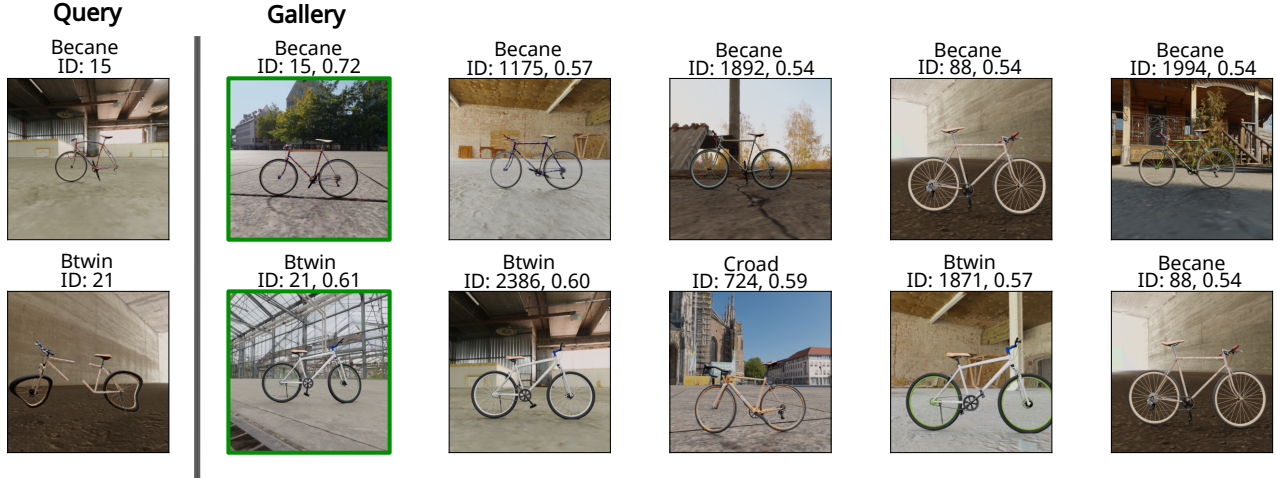


Figure 3: Retrieval results (Top-5 images) for the BL network (ID and similarity scores). The correct ID is retrieved despite the presence of missing parts (ID 15), bent (ID 15) or broken (ID 21) frame, deformed wheels (ID 21), and rust (ID 21).

t-SNE plots of the  $[cls]$  token extracted from the backbone (Appendix C) show only partial overlap between the real and synthetic domains. Saliency (attention) maps generated following the approach in [6] highlight how the network correctly focused its attention on the bike frame (and occasionally the wheels) (Figure 22). Different training regimes consistently yield similar visual keys (Appendix C).

**How does the network generalize to previously unseen bike models?** Overall, the DD task generalizes well to previously unseen models, while performance is more dependent on the specific type of damage. When training on synthetic data alone (BL), we observed an increase in performance for the DD task from  $92.1 \pm 0.2$  to  $94.1 \pm 0.2$  (Table 1). Again, forcing the network to improve on real images lowers the performance on synthetic images for all strategies but BL + PADA ( $94.2 \pm 0.4$ ). However, the latter incorporates an additional bike model classification task, which may help TransReI3D to better generalize to previously unseen models. On the other hand, in the ReID task both TransReI3D and RRT struggle to generalize to completely novel bike models, with a moderate decrease in performance both in terms of mAP ( $79.3 \pm 0.1$  vs.  $85.3 \pm 0.2$ ) and CMC-1 ( $72.5 \pm 0.2$  vs.  $79.8 \pm 0.5$ ).

## 7. Conclusions

In this work, we introduced the novel task of damaged object re-identification. As a benchmark for this task, we introduced the synthetic BBBicycles dataset which contains paired images of the same bike with and without damages. As a baseline, we proposed TransReI3D, a multi-task transformer-based architecture for joint DD and ReID. Experimental results showed how the DD task improves per-

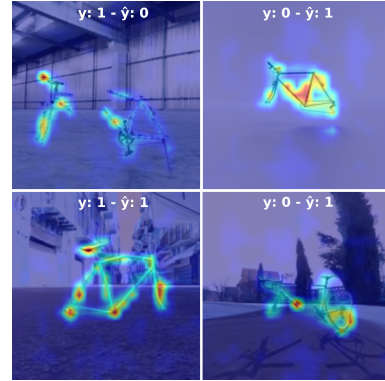


Figure 4: Attention maps of TransReI3D for BL + REAL + DANN, with Bent frame labels ( $y$ ) and predictions ( $\hat{y}$ ).

formance on the ReID task, but not viceversa. The main limitation of the present work is the lack of real paired images of bikes, before and after damage; for this reason, only the DD task was analyzed for real images. As collecting such a dataset would be prohibitively expensive, an option to be explored is simulation, e.g., through data augmentation or generative models. Given the novelty of the task, there is ample room for future expansion in several directions. First, concerning the ReID task, the ability to generalize to previously unseen models should be improved. Experiments should also be extended to include more traditional convolutional architectures. Second, techniques for bridging the synthetic-to-real gap could be further investigated, e.g. by looking at the few-shot and partial/universal domain adaptation literature. Third, segmentation could be leveraged to improve foreground/background differen-



tiation. Finally, other tasks could be explored using the proposed pipeline and the collected 3D models in combination with rendered images, e.g., cross-modal image retrieval [20, 53], segmentation, and 3D part recognition [58].

## Acknowledgements

The authors gratefully acknowledge the financial support of Reale Mutua Assicurazioni.

## References

- [1] Blender. <https://www.blender.org/>. Accessed: 2022-08-29.
- [2] Lily surface scraper. <https://github.com/eliemichel/LilySurfaceScraper>. Accessed: 2022-08-29.
- [3] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. Large-scale instance-level image retrieval. *Information Processing & Management*, 57(6):102100, 2020.
- [4] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018.
- [5] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3D model repository. *arXiv preprint: 1512.03012*, 2015.
- [6] Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 397–406, 2021.
- [7] Wei Chen, Yu Liu, Weiping Wang, Erwin Bakker, Theodoros Georgiou, Paul Fieguth, Li Liu, and Michael S Lew. Deep image retrieval: A survey. *arXiv preprint arXiv:2101.11282*, 2021.
- [8] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet. Vision-based navigation of unmanned aerial vehicles. *Control Engineering Practice*, 18:7:789–799, 2010.
- [9] Rita Cucchiara and Matteo Fabbri. Fine-grained human analysis under occlusions and perspective constraints in multimedia surveillance. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 18(1s):1–23, 2022.
- [10] C. R. d. Souza, A. Gaidon, Y. Cabon, , and A. M. Lopez. Procedural generation of videos to train deep action recognition networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2594–2604, 2017.
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [13] N. Dvornik, J. Mairal, and C. Schmid. Modeling visual context is key to augmenting object detection datasets. In *European Conference on Computer Vision*, pages 375–391, 2018.
- [14] Matteo Fabbri, Guillem Brasó, Gianluca Maugeri, Orcun Cetintas, Riccardo Gasparini, Aljoša Ošep, Simone Calderara, Laura Leal-Taixé, and Rita Cucchiara. MOTSynth: How can synthetic data help pedestrian detection and tracking? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10849–10859, 2021.
- [15] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [17] Shuting He, Hao Luo, Pichao Wang, Fan Wang, Hao Li, and Wei Jiang. Transreid: Transformer-based object re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15013–15022, 2021.
- [18] Christof Henkel. Efficient large-scale image retrieval with deep feature orthogonality and hybrid-swin-transformers. *arXiv preprint arXiv:2110.03786*, 2021.
- [19] Yan Huang, Qiang Wu, JingSong Xu, Yi Zhong, and ZhaoXiang Zhang. Clothing status awareness for long-term person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11895–11904, 2021.
- [20] Longlong Jing, Elahe Vahdani, Jiaxing Tan, and Yingli Tian. Cross-modal center loss for 3d cross-modal retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3142–3151, 2021.
- [21] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, A. Torralba, and S. Fidler. Meta-Sim: Learning to generate synthetic datasets. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4550–4559, 2019.
- [22] Osman Semih Kayhan, Bart Vredebregt, and Jan C van Gemert. Hallucination in object detection—a study in visual part verification. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 2234–2238. IEEE, 2021.
- [23] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, 2021.
- [24] Sultan Daud Khan and Habib Ullah. A survey of advances in vision-based vehicle re-identification. *Computer Vision and Image Understanding*, 182:50–63, 2019.
- [25] M. Khodabandeh, H. R. V. Joze, I. Zharkov, and V. Pradeep. DIY human action dataset generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, page 1529–152910, 2018.

- [26] Jun Li, Bo Yang, Wankou Yang, Changyin Sun, and Hong Zhang. When deep meets shallow: subspace-based multi-view fusion for instance-level image retrieval. In *2018 IEEE International Conference on Robotics and Biomimetics (RO-BIO)*, pages 486–492. IEEE, 2018.
- [27] Pei Li, Bingyu Shen, and Weishan Dong. An anti-fraud system for car insurance claim based on visual evidence. *arXiv preprint arXiv:1804.11207*, 2018.
- [28] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, 2014.
- [29] Shengcai Liao and Ling Shao. Transmatcher: Deep image matching through transformers for generalizable person re-identification. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- [30] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.
- [31] Xinchun Liu, Wu Liu, Tao Mei, and Huadong Ma. Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance. *IEEE Transactions on Multimedia*, 20(3):645–658, 2017.
- [32] Carlo Masone and Barbara Caputo. A survey on deep visual place recognition. *IEEE Access*, 9:19516–19547, 2021.
- [33] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
- [34] J. McCormac, A. Handa, S. Leutenegger, , and A. J. Davison. Scenenet RGB-D: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation? In *IEEE International Conference on Computer Vision*, pages 2697–2706, 2017.
- [35] Lia Morra and Fabrizio Lamberti. Benchmarking unsupervised near-duplicate image detection. *Expert Systems with Applications*, 135:313–326, 2019.
- [36] L. Morra, F. Manigrasso, and F. Lamberti. SoccER: Computer graphics meets sports analytics for soccer event recognition. *SoftwareX*, 12:100612, 2020.
- [37] S. I. Nikolenko. *Synthetic Data for Deep Learning*. Springer, 2021.
- [38] X. Peng, B. Sun, K. Ali, and K. Saenko. Learning deep object detectors from 3D models. In *IEEE International Conference on Computer Vision*, pages 1278–1286, 2015.
- [39] Yifan Peng, Qingyu Chen, and Zhiyong Lu. An empirical study of multi-task learning on bert for biomedical text mining. In *Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing*, pages 205–214, 2020.
- [40] E. Perot, M. Jaritz, M. Toromanoff, and R. d. Charette. End-to-end driving in a realistic racing game with deep reinforcement learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 474–475, 2017.
- [41] Lyle Regenwetter, Brent Curry, and Faez Ahmed. BIKED: A dataset for computational bicycle design with machine learning benchmarks. *Journal of Mechanical Design*, 144(3), 2022.
- [42] V. Vineet S. R. Richter, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision*, 2016.
- [43] P. Schneider and G. Schneider. De novo design at the edge of chaos. *Journal of Medicinal Chemistry*, 59:9:4077–4086, 2016.
- [44] Charu Sharma, Siddhant R Kapil, and David Chapman. Person re-identification with a locally aware transformer. *arXiv preprint arXiv:2106.03720*, 2021.
- [45] Fei Shen, Yi Xie, Jianqing Zhu, Xiaobin Zhu, and Huanqiang Zeng. Git: Graph interactive transformer for vehicle re-identification. *arXiv preprint arXiv:2107.05475*, 2021.
- [46] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, J. Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, 2017.
- [47] Xiujuan Shu, Ge Li, Xiao Wang, Weijian Ruan, and Qi Tian. Semantic-guided pixel sampling for cloth-changing person re-identification. *IEEE Signal Processing Letters*, 28:1365–1369, 2021.
- [48] P. Solovev, V. Aliev, P. Ostyakov, G. Sterkin, E. Logacheva, S. Troeshstov, R. Suvorov, A. Mashikhin, O. Khomenko, and S. I. Nikolenko. Learning state representations in complex systems with multimodal data. *arXiv preprint: 1811.11067*, 2018.
- [49] Fuwen Tan, Jiangbo Yuan, and Vicente Ordonez. Instance-level image retrieval using reranking transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12105–12115, 2021.
- [50] Zheng Tang, Milind Naphade, Stan Birchfield, Jonathan Tremblay, William Hodge, Ratnesh Kumar, Shuo Wang, and Xiaodong Yang. Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 211–220, 2019.
- [51] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.
- [52] J. Tremblay, B. Sundaralingam T. To, Y. Xiang, D. Fox, and S. T. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning*, 2018.
- [53] Mikaela Angelina Uy, Jingwei Huang, Minhyuk Sung, Tolga Birdal, and Leonidas Guibas. Deformation-aware 3d model embedding and retrieval. In *European Conference on Computer Vision*, pages 397–413. Springer, 2020.
- [54] W. Y. Wan and D. Yang. That’s so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors us-



- ing petpeeve tweets. In *Empirical Methods in Natural Language Processing*, pages 2557–2563, 2015.
- [55] Q. Wang, J. Gao, W. Lin, and Y. Yuan. Learning from synthetic data for crowd counting in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8190–8199, 2019.
  - [56] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 79–88, 2018.
  - [57] Cameron R Wolfe and Keld T Lundgaard. Exceeding the limits of visual-linguistic multi-task learning. *arXiv preprint arXiv:2107.13054*, 2021.
  - [58] Chun-Han Yao, Wei-Chih Hung, Varun Jampani, and Ming-Hsuan Yang. Discovering 3d parts from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12981–12990, 2021.
  - [59] Dominik Zapletal and Adam Herout. Vehicle re-identification for automatic video traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–31, 2016.
  - [60] Guowen Zhang, Pingping Zhang, Jinqing Qi, and Huchuan Lu. HAT: Hierarchical aggregation transformers for person re-identification. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 516–525, 2021.
  - [61] Y. Zhang, W. Qiu, Q. Chen, X. C. Hu, and A. L. Yuille. Unrealstereo: A synthetic dataset for analyzing stereo vision. *arXiv preprint: 1612.04647*, 2016.
  - [62] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.
  - [63] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1116–1124, 2015.
  - [64] Liang Zheng, Yi Yang, and Qi Tian. SIFT meets CNN: A decade survey of instance retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2017.
  - [65] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.
  - [66] Kuan Zhu, Haiyun Guo, Shiliang Zhang, Yaowei Wang, Gaopan Huang, Honglin Qiao, Jing Liu, Jinqiao Wang, and Ming Tang. AAformer: Auto-aligned transformer for person re-identification. *arXiv preprint arXiv:2104.00921*, 2021.

## A. Dataset

### A.1. CGI Pipeline implementation details

In this section we provide additional details on the CGI pipeline used to implement the BBBicycles dataset, and in particular on the transformations that are randomly applied to generate “before” and “after” images.

#### A.1.1 Template rig.

A bike contains many movable elements which need to be positioned (to randomly change the bike pose) or deformed (to simulate damages). In order to randomly apply these transformations, we defined a *template rig* that needs to be adapted to match the given bike model. The template rig is composed of an “armature” (Figure 5a), “lattices” (Figure 5b), and “rail guides”, placed as depicted in Figure 5c.



Figure 5: Template rig adaptation and skinning: (a) armature with bone groups and layers, (b) fitting lattice to wheel meshes, and (c) rail guides positioning.

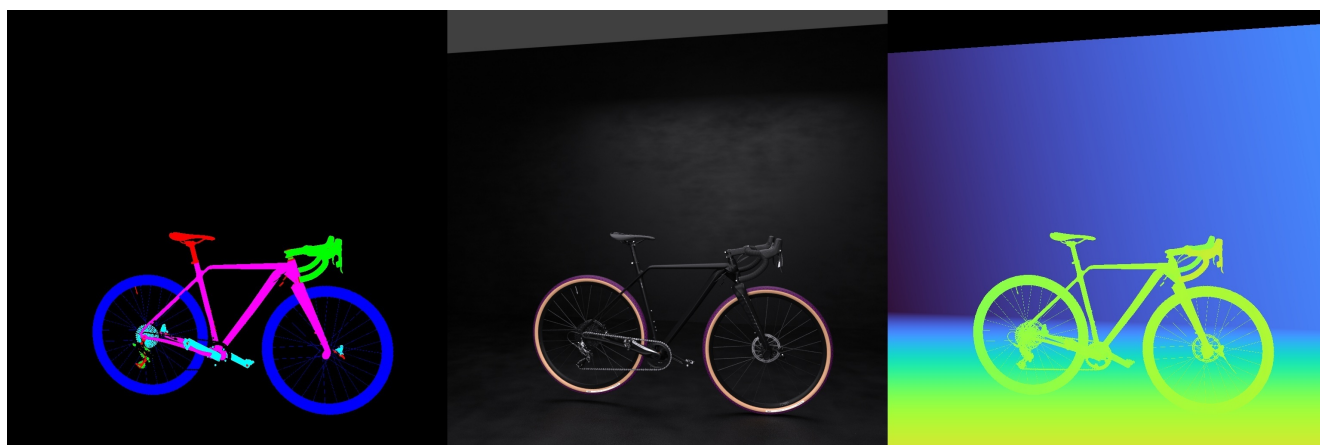
#### A.1.2 Image rendering.

For each ID, multiple images are generated by applying the following transformations:

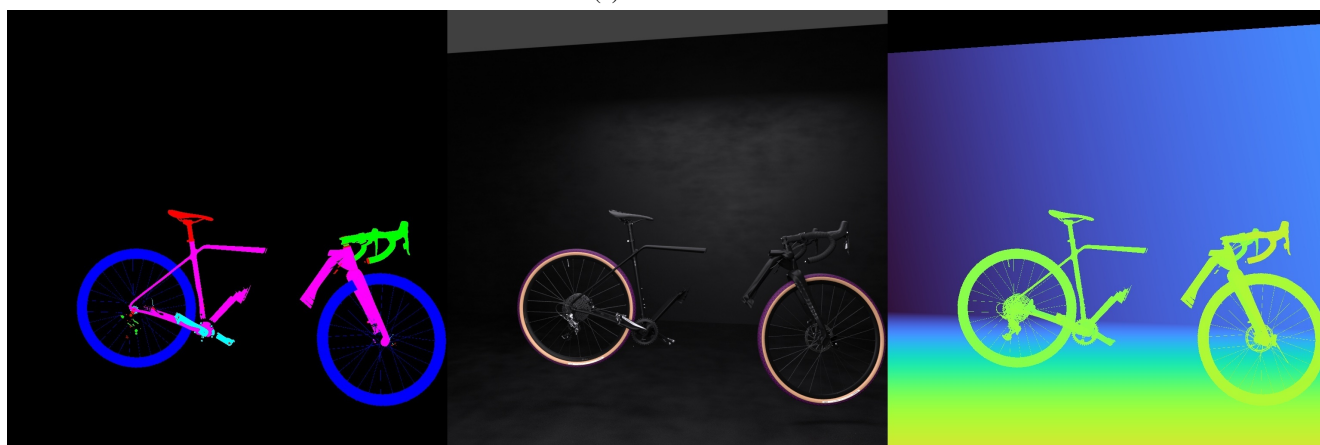
1. Mobile parts composition. This is accomplished by randomly performing one or more actions among: translating the seat and handlebar; rotating the seat, handlebar, pedals, and wheels. The allowed range of movement for each model is set during the rig adaptation.
2. Dirt. A custom shader is used to randomly apply dirt in the form of mud or rust, with a predetermined probability.
3. (Optional) Removing parts. The seat, pedals, handlebar and wheels can be removed with a predetermined probability.
4. (Optional) Damaging parts (frame excluded). Parts of the bikes can be damaged, by selecting a deformation either from the wheels’ library or from the pose library.
5. (Optional) Damaging the frame. The frame can be damaged either by picking a deformation from the pose library (Bent Frame) and/or by breaking it using the rail guides-boolean system (Broken frame). Hence, four possible damage categories are possible: normal frame, bent frame, broken frame, or bent & broken frame.
6. Point of View selection. The virtual camera position is chosen randomly within a given boundary, by randomly switching the visible side of the bike, as well as randomly adjusting the camera focal length within a parametrized range.
7. Environment and Lighting. A combination of background and lighting setup is picked.
8. Segmentation. The bike is segmented in the following classes: “Front Wheel”, “Rear Wheel”, “Seat”, “Crankset”, and “Frame”. Segmentation was implemented using the *bpycv*<sup>1</sup> library. An example of segmentation is shown in Figure 6

---

<sup>1</sup><https://github.com/DIYer22/bpycv>



(a) Intact frame



(b) Broken frame

Figure 6: Examples of the CGI pipeline auxiliary outputs. From right to left: segmentation, rendered image, and depth map.

## A.2. The BBBicycles dataset

In this section we provide additional details on the generated dataset to better illustrate the variety of models and damages/deformations included in the BBBicycles dataset.

### A.2.1 3D Bike models

BBBicycles contains images generated from 20 3D bike models retrieved from dedicated marketplaces. It includes several variants of popular bikes such as Road, Cruiser and MTB. In particular, it contains 6 MTB, 1 Enduro, 6 Road, 1 Circuit, 1 Gravel, and 5 Cruiser. The list of bike models per category is illustrated in Table 2: each bike model was assigned to either the training, validation, or (stress) test set. Examples of renderings from each model are shown in Figure 7.

Table 2: Bike model distribution across BBBicycles in the training, validation and test set. Models marked with (\*) are shared between Train and Validation.

Category	Train	Validation	Test
MTB	mfactory ghost freeride* scalpel*	becane btwin	-
Road	rondo verdonga ghost domane* g1* kuota*	croad	-
Cruiser	oldbike holland* huffy* vintage* wbike*		-
Enduro		-	enduro
Circuit		-	mirage
Gravel		-	gbike

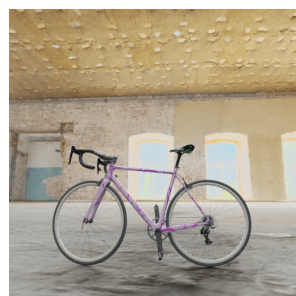




(a) became



(b) btwin



(c) croad



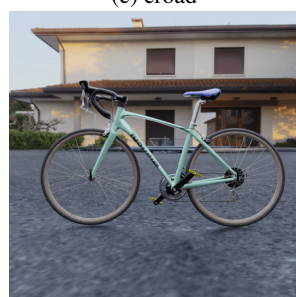
(d) domane



(e) enduro



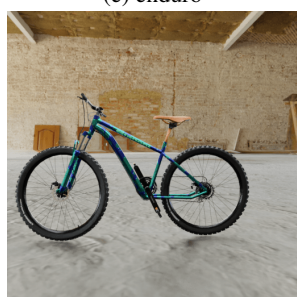
(f) freeride



(g) gl



(h) gbike



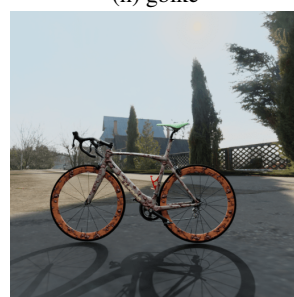
(i) ghost



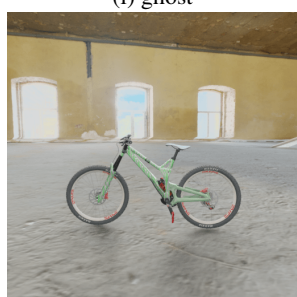
(j) holland



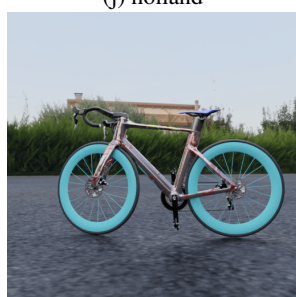
(k) huffy



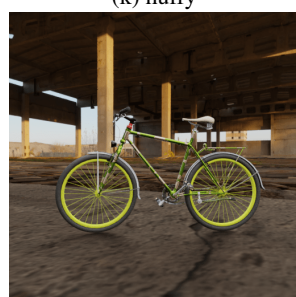
(l) kuota



(m) mfactory



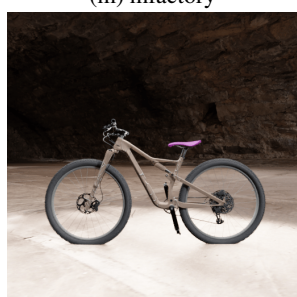
(n) mirage



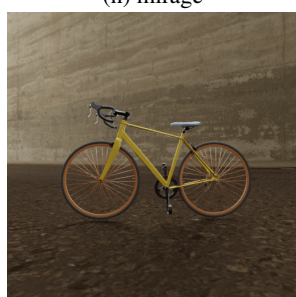
(o) oldbike



(p) rondo



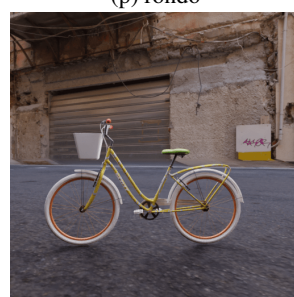
(q) scalpel



(r) verdona



(s) vintage



(t) wbike

Figure 7: Examples of each model used to generate the dataset.

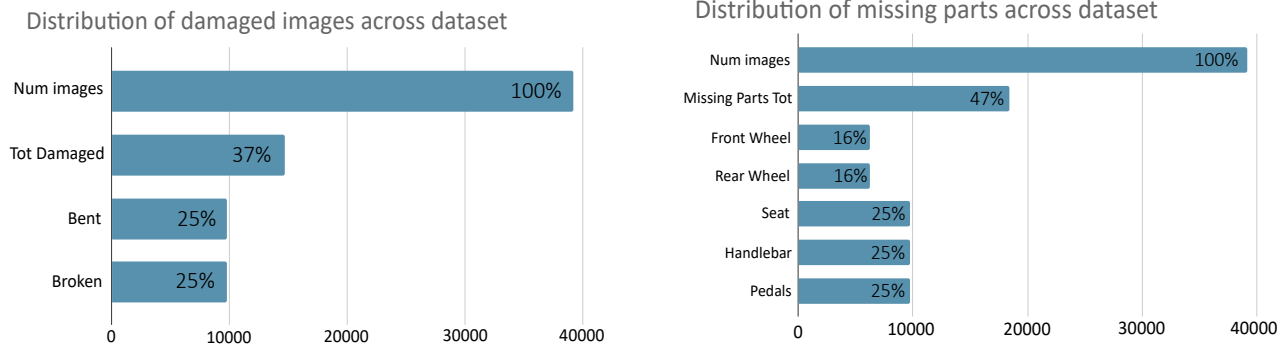


Figure 8: Distribution of damages and missing parts across the synthetic dataset.

### A.2.2 Damage distribution and examples

As illustrated in Section 3, in BBBicycles 50% of the images are generated “before” and 50% “after” a damage occurs. “Before” bikes have a 25% probability of being dirty, while “after” bikes have a 50% chance. “After” bikes are further divided into 25% undamaged and 75% bent, broken or both. As a result, 37% of the total images are damaged (see Figure 8). Examples of damaged and undamaged synthetic bike renderings are shown in Figure 9, Figure 10, Figure 11 and Figure 12.



Figure 9: Examples of bike renderings without damages (to the frame).





Figure 10: Examples of bike renderings with bent frames.



Figure 11: Examples of bike renderings with broken frames.



Figure 12: Examples of bike renderings with bent and broken frames.



Moreover, we set additional labels for each image according to the missing parts of the bike, namely: Front Wheel, Rear Wheel, Seat, Handlebar, Pedals. In the annotations, missing parts are represented by a One-hot vector encoding, where each vector value indicates if the corresponding part is present (0) or not (1), as exemplified in Fig. 15.



Figure 13: One-hot encoding example: the illustrated image only misses the “Rear Wheel”, “Seat” and “Handlebar” parts, hence it has been labeled as “01110”.

### A.3. Real dataset acquisition: additional details.

In this section, we provide additional details on how the real dataset was assembled and annotated.

#### A.3.1 Delft Bikes.

The DelftBikes dataset<sup>2</sup> was originally designed to study whether deep neural networks could hallucinate missing parts in objects. It contains 10,000 bike images with 22 densely annotated parts for each bike. All part locations and part states (i.e., missing, intact, damaged, occluded) are explicitly annotated.

Specifically, we retained only the images from the DelftBikes training set with complete annotations (for some images missing parts annotations were not available), for a total of 8,000 images. Then, we translated the Delftbikes annotations to be compatible with the synthetic dataset annotations, as follows:

- For Front Wheel, Rear Wheel and Seat, we labeled the part as missing if the corresponding part was labeled in the same way (i.e., object state class = missing) in the Delftbikes dataset.
- For Handlebar, we labeled the part as missing if all parts belonging to the group {back handle, front handle, back hand break, front hand break, steer} were also labeled in the same way (i.e., object state class = missing) in the Delftbikes dataset.

<sup>2</sup>[https://data.4tu.nl/articles/dataset/DelftBikes\\_data\\_underlying\\_the\\_publication\\_Hallucination\\_In\\_Object\\_DetectionA\\_Study\\_In\\_Visual\\_Part\\_Verification/14866116](https://data.4tu.nl/articles/dataset/DelftBikes_data_underlying_the_publication_Hallucination_In_Object_DetectionA_Study_In_Visual_Part_Verification/14866116).

- For Pedals, we labeled the part as missing if both parts in the group {front\_pedal, back\_pedal} were also labeled in the same way (i.e, object state class = 2) in the Delftbikes dataset.

None of the bike instances in the DelftBikes dataset presented damages to the frame.

### A.3.2 Web scraping details

We collected samples of real damaged bikes by querying popular search engines (i.e., Google, Bing) and online forums (i.e., Reddit and other dedicated forums). We used different keywords (i.e., “damaged bike”, “bici danneggiata”, etc.) in different languages (i.e., English, Italian, Spanish, French, etc.) in order to increase the number of matches. In particular, we selected countries with higher bike usage like the Netherlands and Denmark. Synonyms of damage were searched to amplify the number of returned images (for instance, “broken bike” and “damaged bike” produce different search results). Additional images of normal bikes were retrieved from second-hand e-commerce sites.

For each scraped image, the origin URL has been serialized as a source reference. The results have then been pruned from unrelated (e.g., excluding images about bike helmets, cycling suits, etc.) and duplicated images by hand and by means of automatic de-duplication techniques, respectively. In particular, we chose a de-duplication technique based on pre-trained CNNs, which marks as duplicated images those with a pairwise similarity score above a given threshold value (experimentally set to 85%).

### A.3.3 Labelling.

All images were manually labeled indicating the damage type and missing parts. Labels were assigned as uniformly as possible to the synthetic dataset. Concerning damage labeling, we set four different labels based on the type of damage present on the bike frame:

- normal: the bike frame is intact, regardless of the condition of the other parts of the bike (e.g., missing parts, damaged wheels, damaged seat).
- bent: the bike frame is bent or presents damage, but it is broken in multiple pieces.
- broken: the bike frame is broken and clearly divided in pieces, and each piece does not present any bending.
- bent & broken: the bike frame is broken and the frame pieces show signs of bending.

For missing parts, we follow the same convention of the synthetic dataset and set additional labels for the following parts: Front Wheel, Rear Wheel, Seat, Handlebar, Pedals. Missing parts are represented by a One-hot vector encoding, where each vector value indicates if the corresponding part is present (0) or not (1), as exemplified in Fig. 15.

## B. Experimental settings

*classification [cls] embedding*, which encodes image global features, is prepended to the sequence of  $N$  *patch tokens*. Each token is encoded by the combination (sum) of the corresponding patch embedding, learnable positional embedding, and SIE embeddings. The  $N + 1$  input tokens, inclusive of the [cls] token for a total size of  $[1, N + 1, 768]$ , are then input into the transformer backbone.

- **Shared ViT Network:** a ViT-like structure including  $L - 1$  layers ( $L = 12$ ) is used as a shared backbone, whose output is then passed to each task-dedicated branch, each including an additional separate  $L^{th}$  transformer layer; each layer attention module has 12 attention heads.
- **ReID global branch:** the ReID task is performed based on the [cls] token alone (which is a global representation of the image features). The token is first passed through a Batch Normalization (BN) layer, whose output is first used for the triplet loss calculation and then passed to a FC layer for performing the ID cross-entropy loss computation.
- **Jigsaw Branch and Jigsaw Patch Module:** in the Jigsaw branch, the Jigsaw Patch Module (JPM) module is applied on the output of the  $L - 1$  shared transformer layers: first the [cls] token is separated from the output of the  $L - 1$  layers, while the remaining part of the output, consisting only of the patch tokens, is randomly rearranged into four equally  $N/4$  sized groups. Then, the previously extracted [cls] token is added to each group so obtained, and each group is

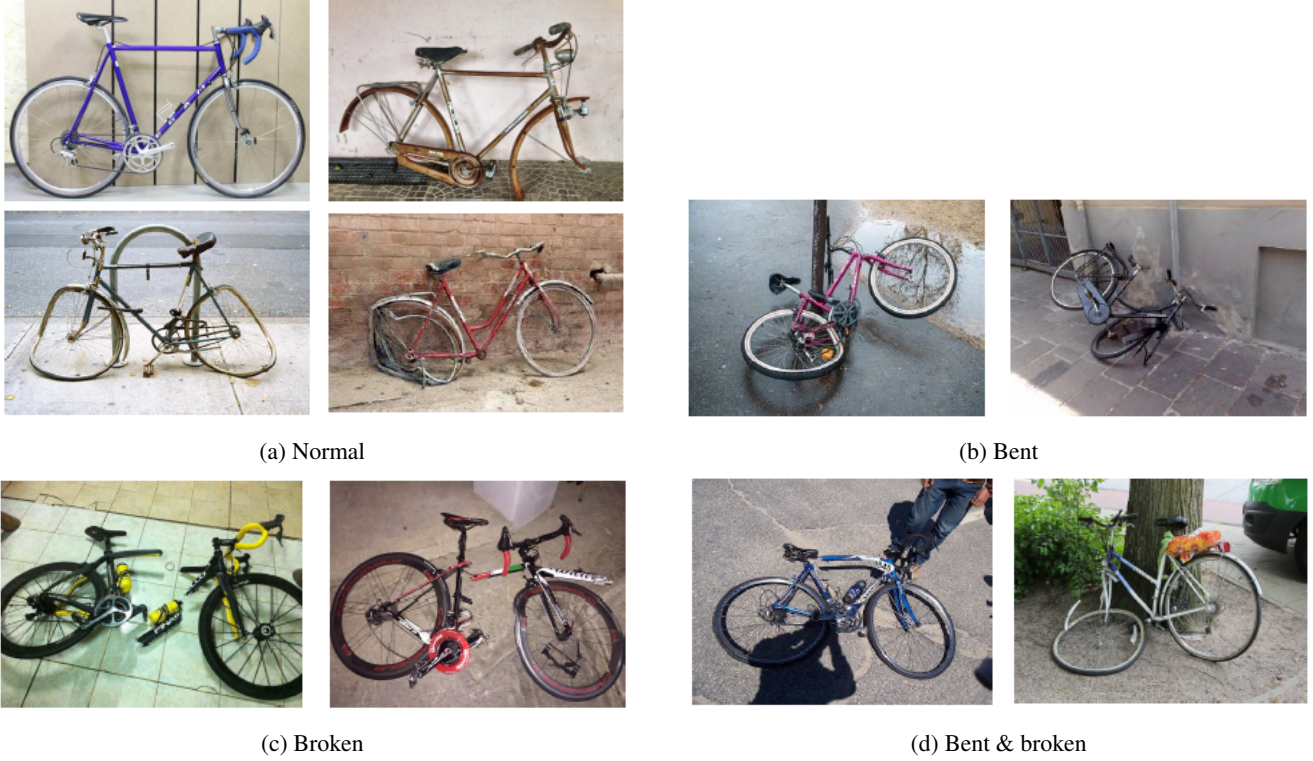


Figure 14: Examples of real images with normal (a), bent (b), broken (c) and bent & broken (d) frames.



Figure 15: Example of real image with missing parts: the illustrated image only misses the “Front wheel” and “Seat” parts, hence it has been labeled with “10100”.

finally passed to  $L^{th}$  transformer layer of the JPM branch; the output of the JPM branch is a set of classification tokens, one for each group. In the same way as in the global branch, each output  $[cls]$  token is passed through a corresponding BN layer, whose output is used for the triplet loss calculation and then passed to the corresponding FC layer for the ID cross-entropy loss. These loss components are added to the combined loss of the global branch to be minimized. In this way, the ReID model learns more discriminative parts and becomes more robust with respect to perturbations.

- **Damage branch:** like for the ReID task, the damage classification is performed on the  $[cls]$  token alone. The token is first passed through 7 different BN layers (one per head), and each output is passed to a corresponding FC layer, one for

Bend frame classification, one for Broken frame classification, and one for each missing part classification. The scores and cross-entropy losses produced by each of these heads are then combined by weighted averaging for the final damage loss.

### B.1. Domain adaptation

The resulting architecture configurations after the addition of DANN and PADA are depicted in Fig.16 and Fig.17, respectively.

Parameters used for domain adaptation are:

- $\theta = 1.0$  as weight for the domain discriminator loss  $\mathcal{L}_{dmn}$  (Equation 3).
- $\delta = 1.0$  as weight for the model classification loss  $\mathcal{L}_{mdl}$ , when PADA is active, otherwise 0 (Equation 3).
- Gradient Reversal Layer weight  $\iota = 1.0$  in all DANN and PADA experiments except for the Base + Real + DANN experiment, in which  $\iota = 10.0$  (Equation 3).

$$\mathcal{L}_{D_{tot}} = \mathcal{L}_D + \theta \mathcal{L}_{dmn} + \delta \mathcal{L}_{mdl} - \iota \frac{\partial \mathcal{L}_{DMN}}{\partial f_g} \quad (3)$$

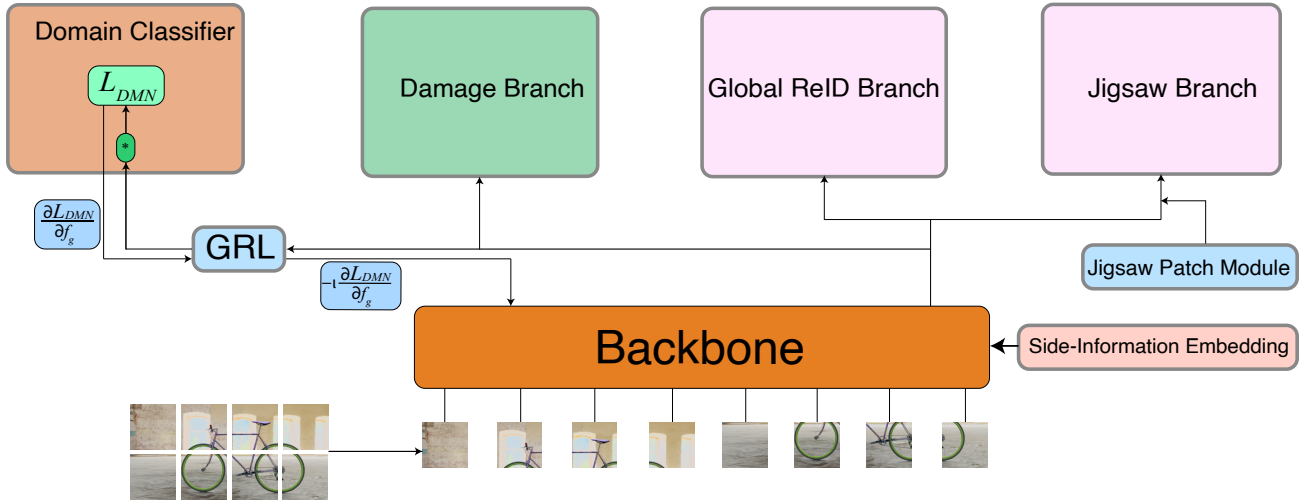


Figure 16: TransReI3D architecture with the addition of DANN components.

## C. Additional results

### C.1. Retrieval examples

Figure 18 depicts some example predictions of TransReI3D on the ReID task.

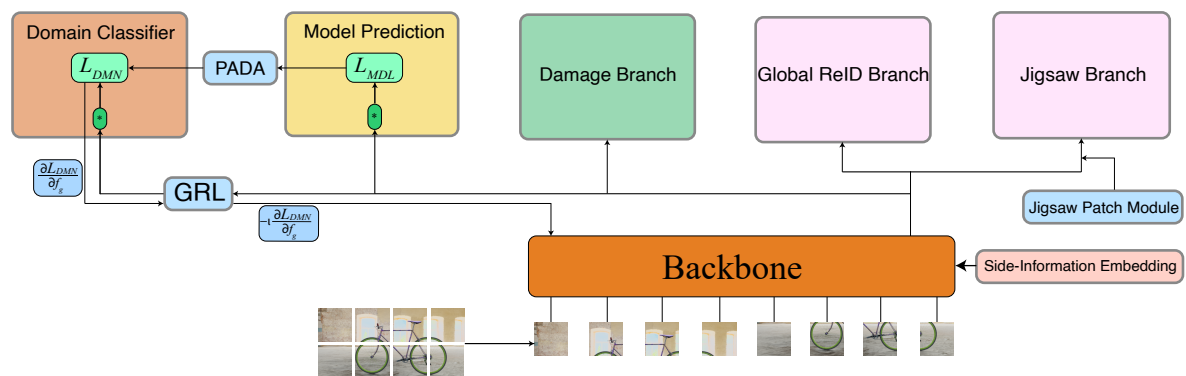


Figure 17: TransReI3D architecture with the addition of the PADA module.



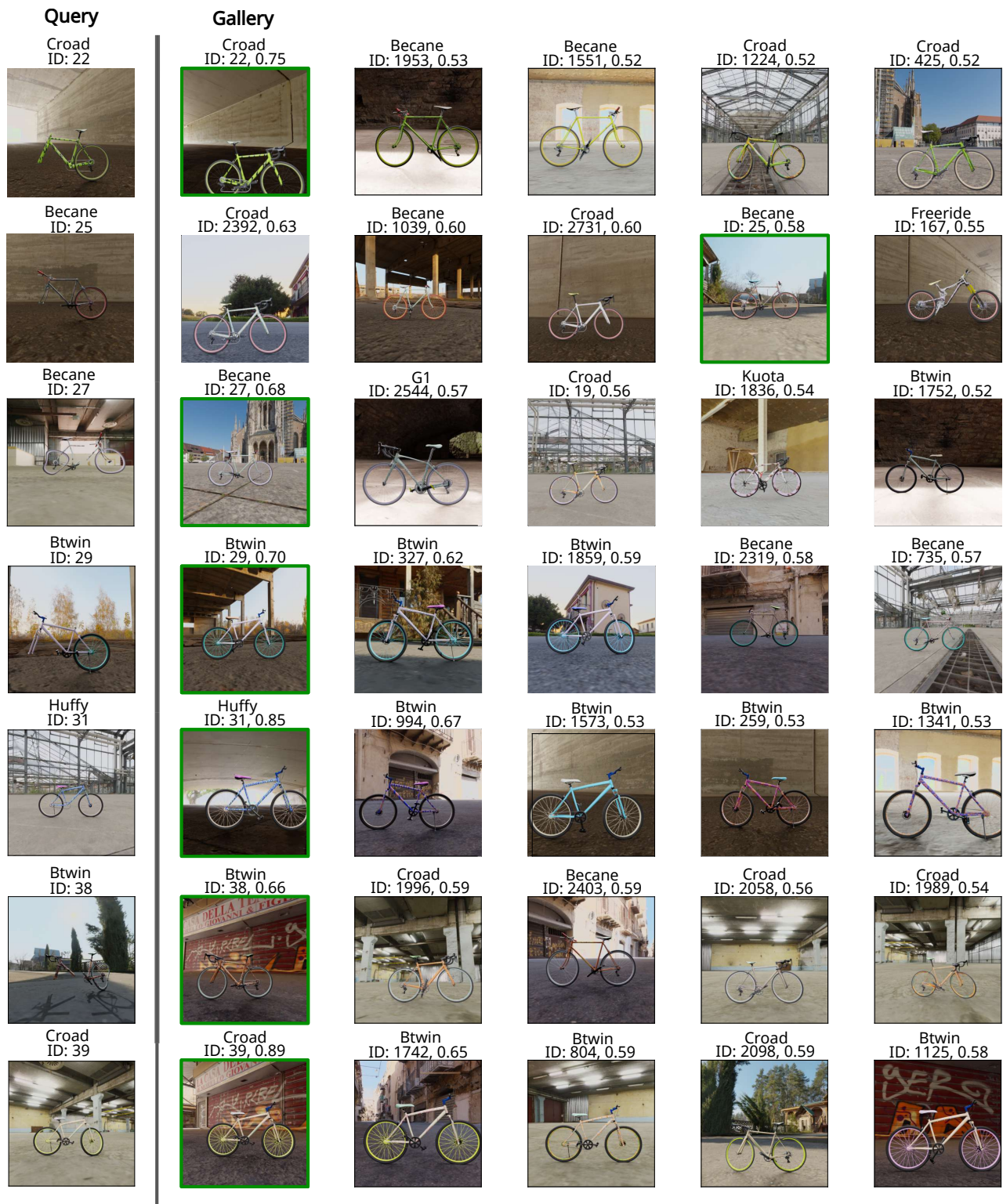


Figure 18: Retrieval results (Top-5 images) for the Baseline configuration, and corresponding model, bike ID, similarity scores. In most cases, the correct result (shown with a green border) is within the Top-5 predictions.

## C.2. Effect of the background on ReID and DD tasks

In this section, we report additional experiments with different backgrounds to understand its effect on the ReID and DD tasks. Specifically, we compared three techniques: (i) the use of HDRI images, with random camera position, to generate the background, as detailed in Section 3; (ii) randomly picking an image from Places365 as background, and (iii) the use of a simple uniform background. To generate samples with Places365 images as backgrounds, we leveraged the original pipeline to render an auxiliary image with a transparent background and overlay it over the photo taken from the dataset. Examples are shown in Figure 19. It should be noticed that the proposed pipeline leverages a limited number of 360° HDRI maps, and even if the proposed pipeline can generate a virtually infinite number of backgrounds by varying the bike position, camera and illumination, the backgrounds will be visually correlated. On the other hand, Places365 contains a much wider range of scenes, but since the bike is randomly positioned, the resulting blend is not always realistic, and the foreground and background are not as consistent as with HDRI maps.

In Table 3, we evaluate the ability of TransReID3D to generalize to the real domain in the following transfer scenarios: HDRI  $\rightarrow$  Real, Places365  $\rightarrow$  Real and Uniform  $\rightarrow$  Real. We found moderately better results when employing the HDRI results, although when real images are available at training time, with HDRI yielding a marginal improvement over Places365.

We further assess the ability to transfer across synthetic domains, specifically we evaluate the following scenarios: Places 365  $\rightarrow$  HDRI and Uniform  $\rightarrow$  HDRI. We found that the network generalizes quite well across different strategies to insert the background, as long as it is not uniform. In the latter case, the performance significantly drops as the network is no longer able to separate the bike from the background.

Based on these results, we conclude that the proposed pipeline contains sufficiently varied backgrounds, and the higher consistency improves the generalization capabilities.

Table 3: TransReID3D performance on the validation set with different strategies to generate the background. The network was trained on synthetic data except for  $\dagger$  (labeled real images available at training time) and  $\ddagger$  (unlabelled real images available at training time).

	Validation					
	Damage Detection		Re-identification (Synthetic)			
	Real AUC	Synthetic AUC	mAP	CMC-1	CMC-5	CMC-10
BG HDRI + Real $\dagger$	<b>97.3 <math>\pm</math> 2.2</b>	<b>91.4 <math>\pm</math> 0.2</b>	<b>85.3 <math>\pm</math> 0.2</b>	<b>79.4 <math>\pm</math> 0.1</b>	<b>92.9 <math>\pm</math> 0.4</b>	<b>96.6 <math>\pm</math> 0.4</b>
BG Places365 + Real $\dagger$	96.3 $\pm$ 1.9	90.4 $\pm$ 0.2	85 $\pm$ 0	79.0 $\pm$ 0.4	92.8 $\pm$ 0.3	96.3 $\pm$ 0.2
BG Uniform + Real $\ddagger$	95.2 $\pm$ 3.4	87.4 $\pm$ 1.5	48.5 $\pm$ 3.4	39.2 $\pm$ 1.9	59.4 $\pm$ 5.6	66.0 $\pm$ 5.7

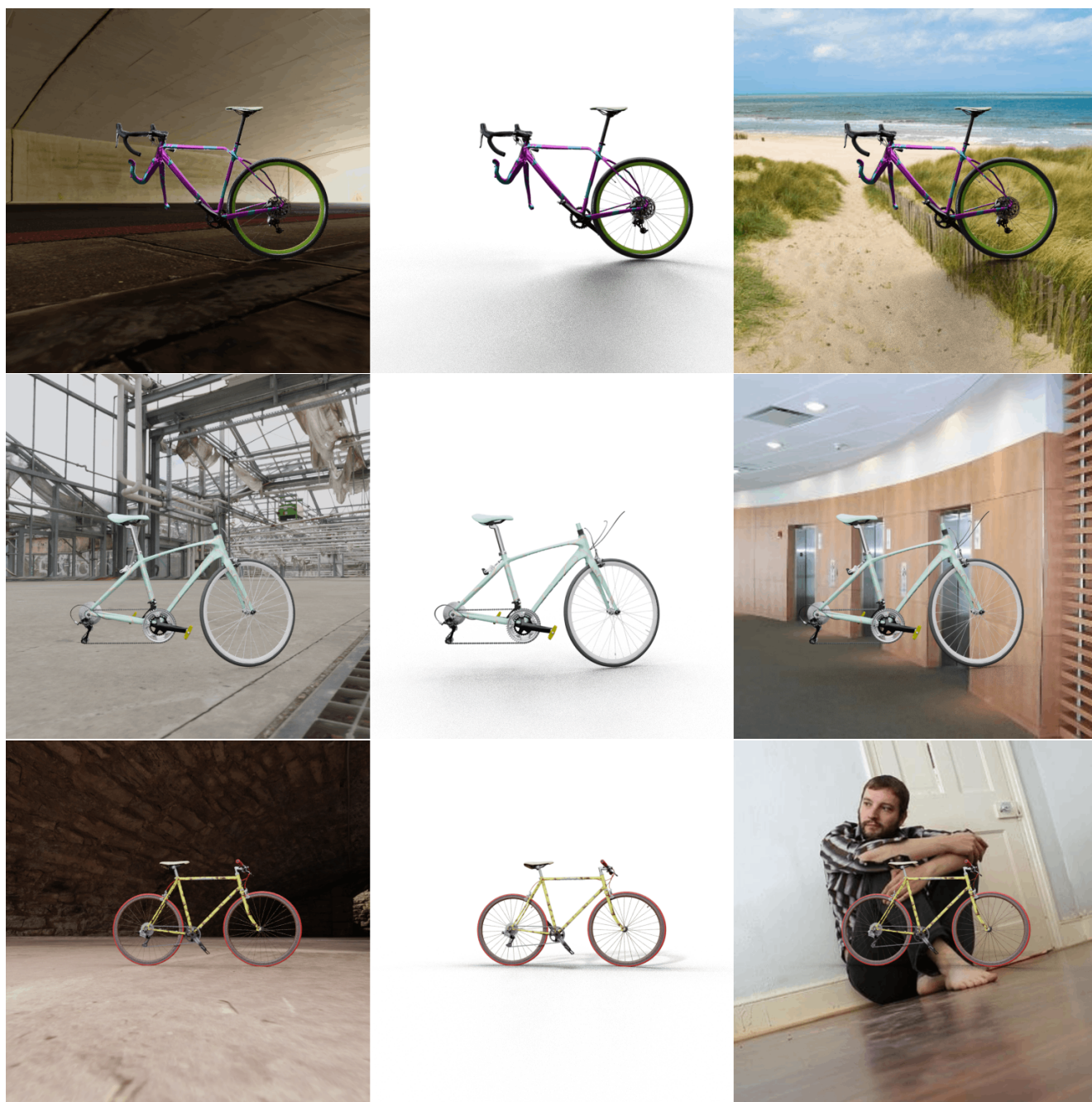


Figure 19: Examples of synthetic bike rendering placed against a 360 HDR background, a uniform background, and random scene from the Places365 dataset.



### C.3. Additional explainability and t-SNE plot

The t-SNE plots of the  $[cls]$  token extracted from the backbone (Figure 20) show partial overlap between the real and synthetic domains, and highlight how real images from various sources yield very different distributions.

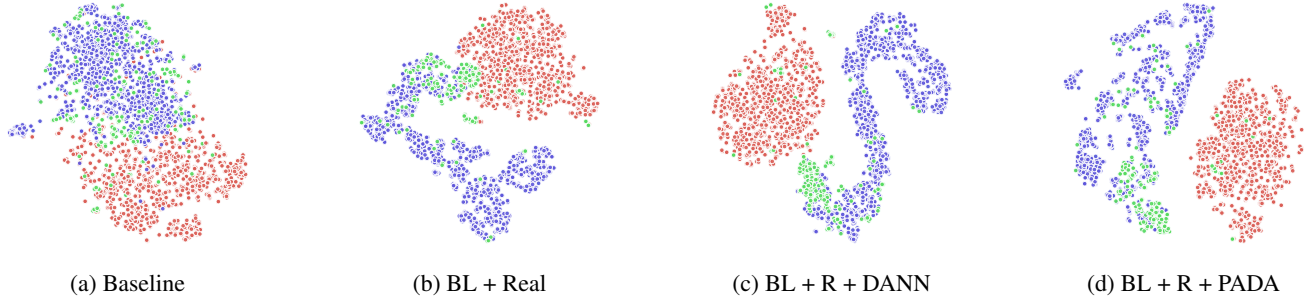


Figure 20: t-SNE plot of the  $[cls]$  token extracted from the DD branch under different training regimes. • DelftBikes (real) • Web scraping (real) • BBBicycles (synth)

The t-SNE plots in Figure 21 illustrate the distribution of the  $[cls]$  tokens from the DD branch and the backbone. By comparing synthetic damaged (red) and undamaged (cyan) examples, it can be seen how the backbone captures features related to the bike model and invariant to the presence of damage, whereas the DD branch clearly distinguishes damaged vs. non-damaged bikes.

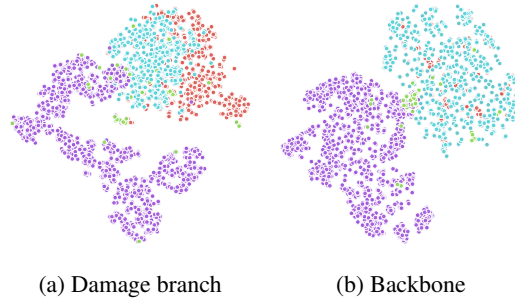


Figure 21: t-SNE plot of the  $[cls]$  token extracted from DD branch (a) and backbone (b) for damaged and non-damaged bike instances (BL + Real setting). • Synthetic no damage • Synthetic damaged • Real no damage • Real damaged

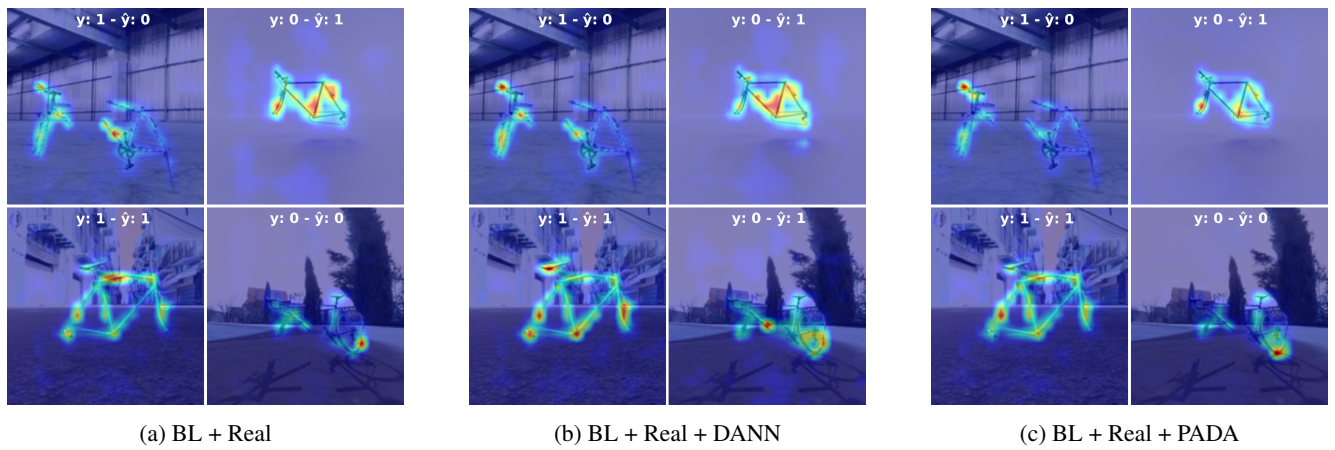


Figure 22: Attention maps of TransRel3D under different training regimes. Values for Bent frame labels ( $y$ ) and predictions ( $\hat{y}$ ) are superimposed on the image.