AbSynth: Using Abstract Image Synthesis for Synthetic Training

Dominik Penk[1,2], Maik Horn[2], Christoph Strohmeyer[2], Bernhard Egger[1], Marc Stamminger[1] and Frank Bauer[1]

[1]*Chair of Visual Computing, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstraße 11, Erlangen, Germany*

[2]*Schaeffler Technologies AG & Co. KG, Industriestraße 1-3, Herzogenaurach, Germany*

Keywords: Synthetic Training Data, Domain Gap, Deep Learning, Computer Vision.

Abstract: We present a novel pipeline for training neural networks to tackle geometry-induced vision tasks, relying solely on synthetic training images generated from (geometric) CAD models of the objects under consideration. Instead of aiming for photorealistic renderings, our approach maps both synthetic and real-world data onto a common *abstract image space* reducing the domain gap. We demonstrate that this projection can be decoupled from the downstream task, making our method an easy drop-in solution for a variety of applications. In this paper, we use line images as our chosen abstract image representation due to their ability to capture geometric properties effectively. We introduce an efficient training data synthesis method, that generates images tailored for transformation into a line representation. Additionally, we explore how the use of sparse line images opens up new possibilities for augmenting the dataset, enhancing the overall robustness of the downstream models. Finally, we provide an evaluation of our pipeline and augmentation techniques across a range of vision tasks and state-of-the-art models, showcasing their effectiveness and potential for practical applications.

1 INTRODUCTION

In modern-day industrial computer vision applications, deep learning, specifically convolutional neural networks (CNNs) (Ciresan et al., 2011), play an important role. Usually, they are trained in a supervised fashion, leveraging annotated datasets of image or video data. These methods routinely outperform humans or hand-crafted approaches on industry-relevant tasks like object classification, detection, or anomaly detection. State-of-the-art models are typically trained on large-scale public datasets, *e.g.*, ImageNet (Isola et al., 2017) or CoCo (Lin et al., 2014). However, these models must be finetuned on a use-case-specific dataset to be used in production. Unfortunately, such datasets are often not publicly available and must be created manually, a time-consuming and costly process prone to errors.

In this paper, we propose a novel pipeline for training or fine-tune neural networks without reliance on real-world data. Our approach requires only a CAD model of the objects of interest and does not rely on realistic rendering like other concurrent work. Furthermore, our method does not require any additional information about surface color or reflectance. This makes our pipeline well-suited for industry-relevant applications, where the shape of the object is the main contributor to derive a solution and CAD models are usually readily available.

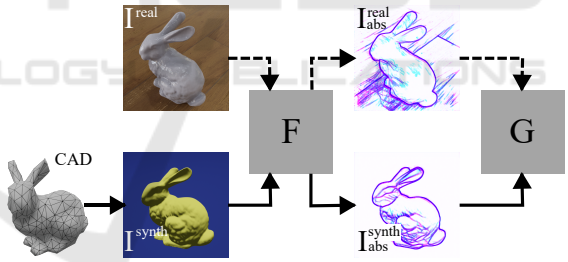


Figure 1: The proposed pipeline projects real and synthetic training images to an abstract representation I_{abs} using a neural network F, which we then pass on to the downstream task G.

The proposed, task-agnostic, **Ab**stract Image **Synth**esis (AbSynth) pipeline is depicted in Fig. 1. The core idea of this pipeline is to transform both real-world and synthetic images into a shared abstract representation I_{abs} using a common transformation F. We will show that F can be fixed for various geometrically based tasks. Consequently, we can then train a network, called G in Fig. 1, in a supervised fashion. At the same time, we will show that the reduced degree of detail leads to straightforward data synthesis

Penk, D., Horn, M., Strohmeyer, C., Egger, B., Stamminger, M. and Bauer, F.
AbSynth: Using Abstract Image Synthesis for Synthetic Training.
DOI: 10.5220/0012431400003660

In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 2: VISAPP*, pages 718-729
ISBN: 978-989-758-679-8; ISSN: 2184-4321

so that no real data or manual labeling is required for training.

2 RELATED WORK

Creating photo-realistic color images is difficult and time-consuming since many user-defined parameters are required. This includes scene parameters, *e.g.*, realistic lighting conditions and camera properties like sensor noise and distortion. All these have a subtle effect, but a network trained on real data might learn some distinguishing features from them. Similarly, a network trained on synthetic RGB images might extract some features from rendering artifacts, *e.g.*, sampling noise on a specific surface material description. Due to this many minute details, the domain gap between real and simulated color images is commonly large.

Narrowing or bridging this gap is an active research topic. Tobin *et al.* were the first to introduce the idea of domain randomization for synthetic RGB data (Tobin et al., 2017). Their work primarily focuses on removing the reliance on accurate surface parameters. The core idea is to force the downstream network to be robust against domain shift by presenting many parameter variations to the network during training. With this approach, the real-world images appear to be merely a different variation. Concretely, they demonstrate that they can train a simple object localizer to identify objects with varying color textures using exclusively synthetic data. Followup work extended this simple idea to other scene parameters, *e.g.*, by adding distractor objects (Tremblay et al., 2018) or random backgrounds (Dosovitskiy et al., 2015).

A different approach called sim-to-real tries to explicitly learn a transformation from the distribution of synthetic to real-world images. This mapping is a general Image-to-Image translation usually realized using a Generative Adverserial Network (GAN) architecture (Goodfellow et al., 2020; Karras et al., 2020). Numerous GAN variations were introduced in recent years, which primarily differ in the data required to train them. For instance, *pix2pix*, introduced by (Isola et al., 2017), uses semantic labels, which are passed to the discriminator and generator networks. The authors show their architecture outperforms color-based GAN architectures. Other approaches rely only on color data from real-world images without semantic labels for domain supervision. One such approach is the *Cycle-GAN* architecture introduced by (Zhu et al., 2017). They also learn the inverse mapping, real-world to synthetic images, and enforce a cycle, from one domain to the other and back, to produce an image similar to the original input.

In recent publications, diffusion models are used to perform Image-to-Image translation (Croitoru et al., 2023).

2.1 Line Drawings

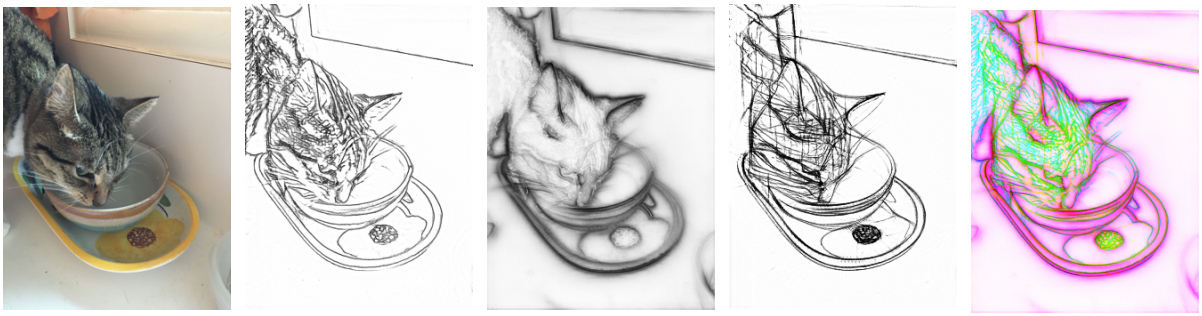
Instead of creating more realistic color images, (Harary et al., 2022) proposed that edge images can be used as the basis for domain generalization. They use edge images to guide a learned *bridge* domain that encapsulates all necessary information for a specified downstream task. They use the bride domain images to force the network, which solves the downstream task, to generalize over multiple input domains, ranging from color images to paintings.

Other studies (Goodman, 2022; Kennedy and Ross, 1975; Hertzmann, 2021a; Hertzmann, 2021b) have shown that edge images convey a strong sense of geometry and can even be used to predict depth images.

Based on these observations, we have chosen line images as the abstract intermediate representation for our AbSynth pipeline. This implies that the function F needs to take an RGB image and output a line drawing of the same image. Usually, an implementation of F only produces a single-channel image. However, as depicted in Fig. 2e, we can combine multiple instances of F to form the final abstract representation I_{abs}. We will show that this combined abstract representation often improves performance.

Edge Detection

Edge detection is a natural fit for F, and since it is a fundamental technique in image processing, many algorithms were developed for this task. One of the most widely used edge detection methods is the Canny edge detection algorithm (Canny, 1986), which involves applying a series of image filters to smooth the image and highlight areas containing large intensity gradients. Unfortunately, these methods expose a couple of user-chosen parameters, *e.g.*, thresholds of intensity differences, which must be chosen carefully to achieve a good edge image. Simple image filters such as Sobel, Prewitt, and Roberts operators can also be used for edge detection. However, these filters often produce noisy or incomplete edges, More recent, data-driven approaches like *Holistically-Nested Edge Detection* (HED), presented by (Xie and Tu, 2015), can be pre-trained and used without the need to set parameters. For completeness, we classify the intermediate bride domain BrAD introduced



(a) Color Input. (b) Anime. (c) BrAD. (d) Open Sketch. (e) Combined.

Figure 2: Different line styles for the same input. Subfibure (e) displays the three styles combined as a single RGB image.

by (Harary et al., 2022) as a version of edge detection since it uses the same model as HED. The authors also ensured that the resulting bride domain images remained visually similar to the output of the original HED network.

Style Transfer

In the most general form, F is a generic image-to-image transformation. Many data-driven approaches have been developed to facilitate such transformations in recent years. For our approach, we are particularly interested in the field of style transfer. Here, the goal is to input an arbitrary color image and return a new image with the same content in another style. Applications in this domain range from simple image colorization to transforming a photograph into a cubistic drawing. We model F under the style transform paradigm by training a network to produce a given style of line drawings. Some example styles are shown in Fig. 2 and range from artistic anime to technical drawings.

The paper *"Informative Drawings: Learning to generate line drawings that convey geometry and semantics"* by (Chan et al., 2022) presents a method tailored to precisely this problem and is focused on generating line drawings that accurately convey both the geometry and the semantics of the original color image. They developed an updated training pipeline incorporating geometric and semantic consistency between the input color image and the resulting line drawing.

They used an explicit semantic loss function between the input and output images to achieve this. This loss uses the network called CLIP (Contrastive Language-Image Pre-training) (Radford et al., 2021), which computes an embedding vector of an image. The CLIP model was trained on pairs of images and their description such that the embedding captures the most important semantic information. The semantic loss then compares the embeddings of the original color image and the resulting line drawing, forcing the network to produce images with similar semantic information.

However, (Chan et al., 2022) points out that the semantics alone does not ensure a geometrically consistent transformation. For instance, both images could contain a plane, but in one image, it is flying in the air, while it is parked on the ground in the other one. They, therefore, introduce a geometric loss that uses a pre-trained monocular depth estimator network. During training, they assume known depth maps for the color images and use the depth estimator network to compute depths for the generated line drawings. The loss is simply the average per-pixel difference between those two depth maps. Interestingly, the single image depth estimator uses VGG19 features and is pre-trained on color images. This confirms the observation made by (Hertzmann, 2021b) that line drawings can adequately convey geometry.

3 METHOD

3.1 Synthetic Image Generation

We now present our synthetic data generation process for the AbSynth pipeline. Our goal is to provide a simple rendering setup that only requires the geometric data of target objects and produces training images containing enough visual information to be converted to a realistic edge image. The method is inspired by early works of (DeCarlo et al., 2003). They informally introduce *suggestive contours* as those regions on a mesh that are real contours in nearby viewpoints. They also define them more formally using the contour indicator function on a smooth surface S

$$\boldsymbol{n}(\boldsymbol{p})^T \boldsymbol{v}(\boldsymbol{p}) \tag{1}$$

where $\boldsymbol{p} \in S$ is a surface point, $\boldsymbol{n}(\boldsymbol{p})$ is the unit surface normal, and $\boldsymbol{v}(\boldsymbol{p})$ is the view vector from the camera

(a) Contour Indicator. (b) Surface Color. (c) Combined. (d) Instance Mask.

Figure 3: (a)-(C): Shading of synthetic training images combines a contour indicator function and colors based on surface normals. During data synthesis, we can easily generate per pixel label, *e.g.* instance masks in (d).

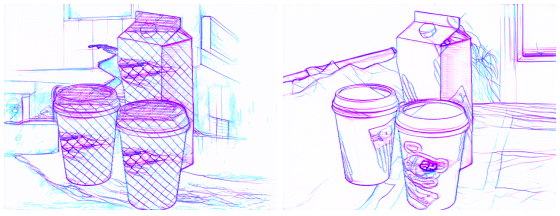
to the surface point. With this, suggestive contours are the local minima of this indicator function, and real contours are found at the roots. DeCarlo *et al.* show that an approximation of these minima can be found by rendering the object and shading the pixels using the dot product from Eq. (1). The resulting images look like Fig. 3a, and intensity ridges are the suggestive contours.

Since the suggestive contour map, defined by Eq. (1) only yields the surface intensity, we can use the hue to provide more information. For this, we adopted an idea from the field of surface normal estimation, where different colored lights are placed along the cardinal direction of a predefined coordinate system. If the object is diffuse and white, a surface directly facing one of these directions will reflect only that light, whereas partially rotated ones will take on a mixed color. We approximate this effect by using the surface normal, mapped from $[-1,1]^3$ to $[0,1]^3$, as the color of the surface, this produces the colors depicted in Fig. 3b. If multiple objects are in the scene, we also apply a random hue shift per object to ensure good visual separation.

3.2 Abstract Augmentation

A common method to improve generalization and prevent shortcut learning is dataset augmentation. Of course, typical image-level augmentations can be applied together with the AbSynth pipeline. Most pixel-wise augmentations like randomized hue shifts or Gaussian blur have a negligible effect since these changes bearly affect F. On the other hand, content-altering methods, *e.g.*, random flipping or image cropping, do not lose any potency in our setup.

Since I^{abs} is still an image, we can use augmentation during training directly on the abstract representation. As we will see in this subsection, the sparse nature of the line images enables augmentations that are impossible or hard to pull off in the original RGB space.



(a) Naive Fusion. (b) Deferred Fusion.

Figure 4: Sample of an abstract training image. The line fusion adds the background. In the deferred approach, lines on the foreground objects follow the surface curvature.

3.2.1 Line Fusion

Remixing multiple images from a dataset to generate new images and add variety is a potent method. A famous example of such a technique is the mosaic data augmentation introduced by (Bochkovskiy et al., 2020), which greatly contributed to improved object detection performance in YOLOv4. However, due to the high complexity of natural images, these approaches are usually rather limited: We can either mix two images using alpha blending or stitch them to form a bigger image.

In contrast, using the sparse nature of our abstract representation enables us to combine the features of two images more easily. Say we want to enrich a training image with distracting features from a different (natural) image J. We first compute J_{abs} using the image-to-image transformation F_{abs}. Then we create a fused image $\hat{I}_{abs}$ by taking the pixel-wise minimum:

$$\hat{I}_{abs} = \min\left(I_{abs}, J_{abs}\right) \tag{2}$$

Since this method essentially merges the lines of two abstract images, we call this augmentation Line Fusion (LF). In Section 4.1, we show that this simple approach can improve generalization since the additional, randomized complexity forces the downstream network G to distill more robust features.

However, the resulting fused images are usually not plausible since the lines of J_{abs} are painted over the original training image without any geometric reasoning. If we control the data synthesis process, or if

the dataset provides more information besides color, we can turn this LF augmentation into a potent content augmentation technique. Usually, any dataset contains images with objects of interest in the foreground. If the dataset provides a per-pixel instance mask for the target objects, we can ensure that they stay in front of the distractor image J by only fusing pixels where the instance mask M indicates the background:

$$\hat{I}_{abs}(\boldsymbol{q}) = \begin{cases} \min\left(I_{abs}(\boldsymbol{q}), J_{abs}(\boldsymbol{q})\right) & \text{if } M(\boldsymbol{q}) = 0 \\ I_{abs}(\boldsymbol{q}) & \text{else} \end{cases} \quad (3)$$

Here, we assume that a value of M is the instance id, 0 being the background. This method allows us to randomize the background, and to place the target objects in varying environments.

Our data generation outlined in Section 3.1 does not use surface textures, leading to featureless surfaces. Real-world objects, on the other hand, often contain non-geometric texture details which impede network performance during inference. We can use LF to counteract this by adding random surface textures to foreground objects using a second distractor image T. We produce the randomized surface texture by fusing T_{abs} with the foreground regions of I_{abs}. We use a random pixel offset $\boldsymbol{\delta}_k$ per object instance to access T_{abs}, which makes sure that the distractor image is broken up, even if many target objects overlap:

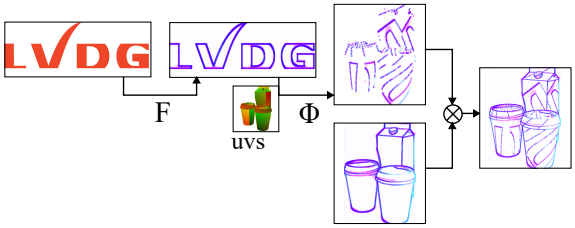
$$\hat{I}_{abs}(\boldsymbol{q}) = \begin{cases} \min\left(I_{abs}(\boldsymbol{q}), J_{abs}(\boldsymbol{q})\right) & \text{if } M(\boldsymbol{q}) = 0 \\ \min\left(I_{abs}(\boldsymbol{q}), T_{abs}(\boldsymbol{q} + \boldsymbol{\delta}_k)\right) & M(\boldsymbol{q}) = k \end{cases} \quad (4)$$

A result of this naive fore- and background fusion approach is depicted in Fig. 4a.

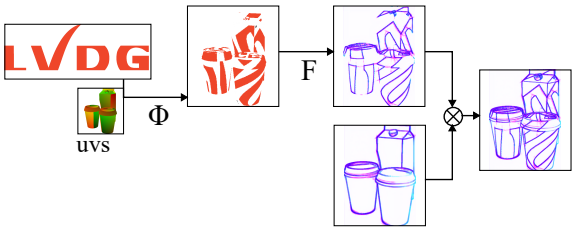
Upon closer inspection of this image, we see that the overall curvature of the coffee cups is hard to understand due to the flat lines pasted over this region. This is contrary to our assumption that the downstream task should be mainly focused on geometric features. To ensure the fused real image follows the actual surface geometry, we borrow an idea of deferred shading and output a uv-mask Φ for the rendered training image. We then use Φ to warp and map T_{abs} along the surface of the target objects:

$$\min\left(I_{abs}(\boldsymbol{q}), T_{abs}(\Delta_k \Phi(\boldsymbol{q}))\right) \text{ for } M(\boldsymbol{q}) = k \quad (5)$$

Here, we also apply a per-instance random coordinate transformation Δ_k to the uv coordinates. In Fig. 5a, we show an example where a logo is warped onto the cups using this approach. The mapped lines are thinner and less visible than the naive warping method. This aliasing effect is caused by the mapping of the (potentially) large and very sparse line image T_{abs} onto a comparatively small region. We can address this problem by applying the image warping to T before we apply F_{abs} and then using Eq. (4) with a constant $\delta_k = 0$. Applying this early-warping method to the logo of the example, we obtain the new foreground distractor depicted on the lower left in Fig. 5a. The resulting foreground fusion on the right is much cleaner, with even line thickness and brightness.



(a) Late warping.



(b) Early warping.

Figure 5: Difference between warping the abstract lines (a) versus warping the distractor image (b). Warping lines produces to magnification and minification artifacts. In contrast, early warping produces clean synthetic images.

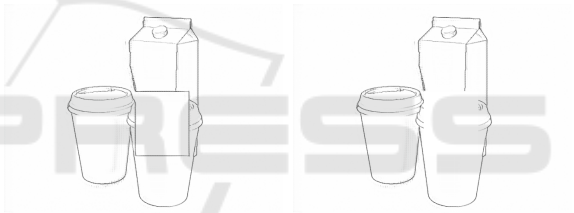


Figure 6: Abstract representation with random erase augmentation. On the left, the augmentation was performed on the color image. On the right on in the abstract domain.

3.2.2 Random Erase

Random erasing, introduced by (Zheng et al., 2021), is a popular augmentation method for object detection and instance segmentation to simulate object occlusion. This augmentation selects a random subregion of the input image and fills it with a constant value. We tried to apply this augmentation to the intermediate representation directly onto the abstract image I_{abs}. As shown in Fig. 6, the results differ considerably depending on the time of content removal. Applying the erase augmentation to the color image

Table 1: Comparison of classification for a ResNet34 trained on synthetic datasets and evaluated on a subset of CoCo without and with the AbSynth pipeline.

Dataset	Style	Augmentation		Accuracy
		LF	RE	
VisDA 2017	Baseline			0.363
	Ours (BrAD)	✗	✗	0.429
		✓	✗	**0.566**
		✓	✓	0.547
AbsDA	Baseline			0.288
	Ours (BrAD)	✗	✗	0.430
		✓	✗	**0.592**
		✓	✓	0.488

yields a visible rectangle (that occludes the object). Erasing regions in the abstract image mimics another observation: Real-world images may contain areas where object boundaries are not visible due to low contrast. Specular reflections, shadows, or similar surface colors may cause this. Since all the presented versions of F rely — at least partially — on intensity gradients, no line will be drawn in these regions. In contrast, our data generation usually has high contrast leading to images with clearly defined object boundaries. We assumed, that random line erasing on the abstract image might improve the downstream network for some datasets. As we will see, this was not the case for our experiments.

4 EXPERIMENT RESULTS

4.1 Classification

In this section, we demonstrate synthetic training for image classification. To this end, we use the VisDa 2017 (Peng et al., 2018) dataset, which consists of rendered training images — with model taken among others from ShapeNet (Chang et al., 2015) — and test data selected from CoCo (Lin et al., 2014). Since instance and uv-masks are not provided with the ViDA dataset, we can only apply the naive version of LF.

To assess the impact of deferred LF, we created a separate dataset called AbsDA. We only took objects from ShapeNet Core, which drops the number of classes in the dataset to 8. Some samples from the two train datasets are depicted in Fig. 7, and the last row shows some examples from the test dataset.

We evaluate the classification accuracy of a ResNet34 model trained on the VisDA 2017 and AbsDa datasets. The results are summarized in Table 1.

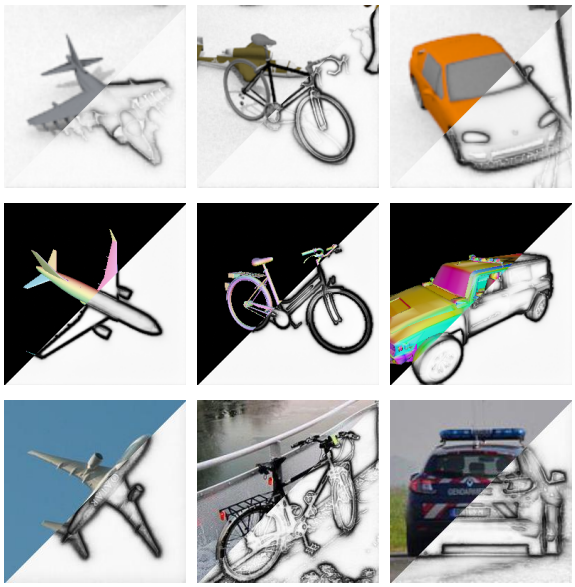


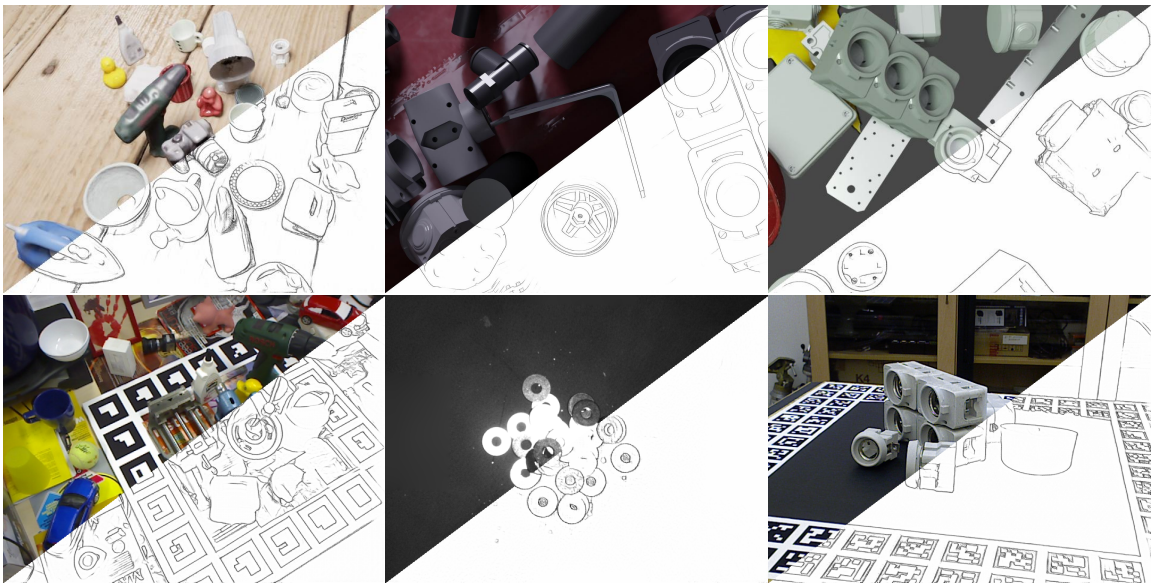
Figure 7: Sample images from different target domains, starting from the top row: VisDa, AbsDa, and CoCo. Each image displays the original color image and its abstract representation (BrAD).

The *Baseline* style is used as a reference point, where no image abstraction or augmentation is applied, and its accuracy acts as a lower bound for the expected performance. For these experiments, we opted to use the BrAD intermediate representation as it was originally designed for cross-domain classification by (Harary et al., 2022). As shown in Table 1, the abstraction greatly improves the classification accuracy, and even the naive LF forces the network to generalize better to real-world data. In contrast, random line erasing (RE) decreases the classification accuracy.

The results of this experiment highlight the importance of using LF as an augmentation method to enhance the generalization ability of models trained using our pipeline. This augmentation introduces additional complexity to the images by adding a background scene. Furthermore, the addition of lines in the region of the foreground objects essentially randomizes their surface texture, which means the model must learn to recognize objects by consistent geometric features and outlines instead of specific textural details.

4.2 Object Detection

Object detection and localization are essential in various visual inspection applications, including quality control and robotic manipulation. To evaluate the performance of our pipeline on a state-of-the-art object detection model, we utilized three publicly avail-



(a) LM-O. (b) ITODD. (c) T-LESS.

Figure 8: Training and evaluation samples from the BOP challenge datasets. Images display the original color input and the corresponding abstract representation.

able datasets from the BOP challenge, namely, LM-O (Brachmann, 2020), ITODD (Drost et al., 2017) and T-LESS (Hodan et al., 2017). As with all datasets in the BOP challenge, they contain synthetic training data generated using BlenderProc (Denninger et al., 2019). BlenderProc uses Blender (Foundation, 2022) and its built-in ray tracing engine to generate physically based renderings of 3D models, which are also included in the datasets. While the training data is synthetic, the test data for each dataset is composed of real-world images, making the evaluation more realistic and relevant to real-world applications.

The three chosen datasets all contain objects with little to no distinguishing textures which implies object detection and classification is primarily geometry based. Besides this similarity, the three datasets present different challenges:

The LM-O dataset features 10 household objects with discriminative shapes, sizes, and colors. Despite the low number of classes, the dataset is challenging for object localization due to the high levels of occlusion and cluttered backgrounds. The object models in the dataset were reconstructed using a depth camera, leading to relatively noisy and inaccurate meshes.

In contrast, the ITODD dataset comprises 28 industrial objects with handcrafted CAD models, resulting in cleaner synthetic training data. The test dataset was captured in a realistic, productive setting using different sensors, including a grayscale camera. The test images were captured against a uniform background with varying levels of inter-object occlusion. For example, some scenes contain a pile of washers, as shown in Fig. 8b while others contain fewer, clearly separated objects.

Finally, the T-LESS dataset combines the features of the previous two datasets. It includes 30 textureless, industry-relevant objects with clean CAD models. The 20 test scenes vary in complexity, with some scenes containing clutter objects. The objects in the dataset exhibit symmetries and mutual similarities in shape and size, and some are composed of other objects, making the dataset more challenging. A sample image from the T-LESS dataset is shown in Fig. 8c.

Realistic Training Data

In this experiment, we aim to evaluate the effect of the proposed abstraction on the training process. To achieve this, we train a Faster-RCNN (Girshick, 2015) on the synthetic (semi-)realistic training data provided by the BOP challenge datasets. We use various intermediate styles, ranging from classic canny-edge images to combinations of line drawing styles. Besides different styles, we also test the impact of the augmentation techniques, presented in Section 3.2, on the final detection quality. Each network was trained for 30 epochs using SGD with a learning rate of 0.02, which we decreased after epochs 16 and 22 by a factor of 10. The Faster-RCNN network uses ResNet50, pre-trained on ImageNet using o ur pipeline, for feature extraction. In Table 2, we report the mean Average Precision (mAP) on the test results for different

Table 2: Detection Accuracy on BOP challenge datasets using the provided synthetic training data. *Baseline* indicates training without the AbSynth pipeline.

Dataset	Style	Augmentation		mAP
		LF	RE	
LM-O	*Baseline*			**0.516**
	Canny	✗	✗	0.423
	BrAD	✗	✗	0.490
	Anime	✗	✗	0.480
		✓	✗	0.495
ITODD	*Baseline*			0.618
	Canny	✗	✗	0.756
		✓	✗	0.783
	BrAD	✗	✗	0.617
	Anime	✗	✗	0.813
		✓	✗	0.795
	Anime + OS	✗	✗	**0.849**
		✗	✓	0.812
		✓	✗	0.819
T-LESS	*Baseline*			0.195
	Canny	✗	✗	0.677
		✓	✗	0.660
	BrAD	✗	✗	0.290
	Anime	✗	✗	0.695
	Anime + OS	✗	✗	0.685
		✓	✗	**0.695**
		✓	✓	0.658

configurations of the training pipeline.

To provide a baseline for comparison, we also train the network using only the original training dataset without using our pipeline. The results in Table 2 show that the proposed AbSynth pipeline significantly improves detection accuracy compared to the baseline approach. Specifically, using the AbSynth pipeline with the Anime and Open Sketch styles and LF achieves the highest mAP on all three datasets.

Interestingly, the BrAD style underperforms compared to any intermediate representation based on the informative drawing architecture. This contrasts with the previous chapter, where the BrAD style was the best choice for image classification.

To understand this difference, we considered the training procedures for the two styles. The BrAD network was trained by (Harary et al., 2022) to contain cues for cross-domain image classification. While this task may partially use shape-based information, the authors did not explicitly force the network to include geometric information in its representation.

Table 3: mAP and per class precision for a subset of objects in the ITODD dataset. Line fusion augmentation often significantly improves network predictions on a per-class level.

LF	mAP	Box	Cap	Fuse	Wash
✗	0.742	0.789	0.900	0.845	**0.850**
✓	**0.827**	**0.950**	**0.904**	**0.911**	0.764

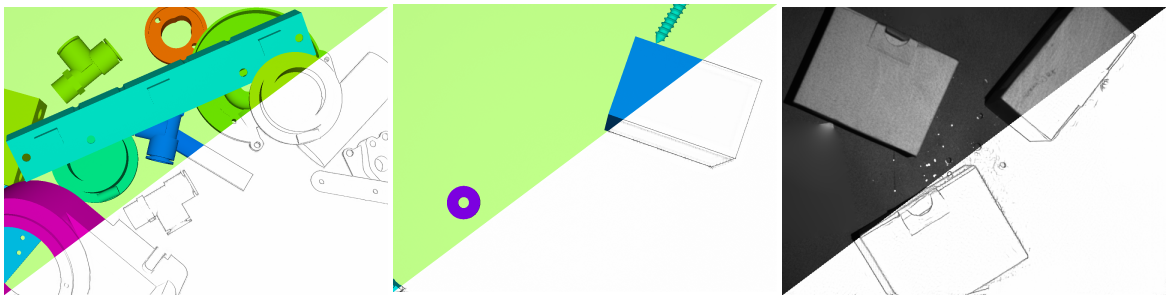
On the other hand, as we discussed in Section 2.1, the line drawing style transform explicitly forces the resulting images to contain geometric details. Since the datasets used in our evaluation contain mostly textureless objects, their geometry becomes the primary feature, and the intermediate style that best conveys it performs the best.

The impact of the RE augmentation on the final detection accuracy was found to be minor and sometimes negative according to the results in Table 2. This technique was designed to increase the network's robustness against partial occlusion, a common challenge in object detection. However, since the training data already includes many examples of inter-object occlusion, the network is trained to deal with partial views of the target objects. On the other hand, the rectangular crop regions used in the augmentation are not commonly found in real-world images, which may lead to a slight increase in the gap between the training and inference domains.

False Color Training Data

We have already shown that our pipeline can improve detection quality for (semi-)realistic synthetic training data. However, to generate such high-quality renderings, the user needs to provide object textures and other surface parameters, e.g., how specular the object is. In Section 3.1, we presented a rendering technique that takes just the CAD model and outputs *false color* images that can be converted to plausible intermediate line representations. To avoid the need to create complex background scenes, we opted to create a custom dataset for ITODD. We created the training data using Blender, with objects randomly scattered on a plane using the built-in rigid-body physics engine. Some samples from the sample data are depicted in Fig. 9.

We again trained a Faster-RCNN network with hyperparameters similar to the previous section. We used Anime line drawings as the intermediate representation since a single-style intermediate representation is a reasonable tradeoff between expected quality and inference speed. In Table 3, we present the detection precision on the test dataset. The mAP is competitive with the network trained using realistic



(a) Multi object train image. (b) Box train image. (c) Box test image.

Figure 9: Synthetic and real-world data for the box class. The images show the original color image and their anime line drawings.

renderings. While the line fusion augmentation was not beneficial on the original ITODD training data, it was for our custom dataset.

We analyze the per-class detection accuracy in more detail to investigate how LF augmentation influences the final network. To this end, we present a selection of per-class detection accuracies in Table 3. In most cases, the augmentation had a minor effect on the results, with only a few classes showing a significant improvement. One example is the packaging box which uses a simple cuboid as its CAD model. As depicted in Fig. 9c, the real-world counterpart has more details, such as a flap to open it. Without augmentation, the network often misclassifies such objects because they lack these details in the training data. However, with LF-augmentation, the network is trained to be robust against varying surface details inside the object. Therefore, the additional details are treated as random surface details and ignored by the trained network, resulting in a more accurate detection.

4.3 Bin Picking

In this section, we apply the proposed AbSynth pipeline to train a model to detect the electric motor shown in Fig. 10a. As depicted in Fig. 10b, we simulate a bin-picking scenario placing the motor in a box alongside other objects.

Similar to the previous section, we used Blender (Foundation, 2022) to generate the synthetic training data. The dataset comprises 600 scenes rendered using 20 random camera positions, yielding 12000 training images. We created a box with random dimensions for each scene and used the built-in physics engine to let objects fall into it. The entire scene generation and rendering procedure was automated, so we only needed to convert the CAD models from the step file format to one that can be imported into Blender. Since we used the false-color rendering approach outlined in Section 3.1 we do not need to create any randomized lighting setup or provide any surface properties.

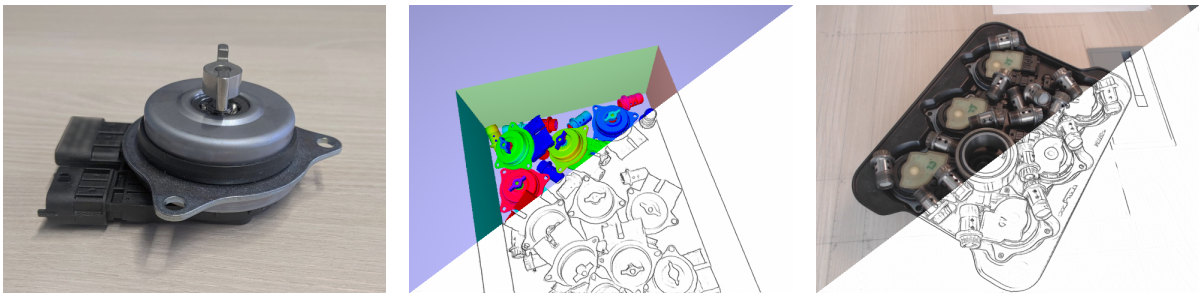
Table 4: Evaluation metrics for custom bin-picking dataset.

	$mAP_{0.5}$	$mAP_{0.75}$	$mAP_{0.5:0.95}$	mAR
Full	0.906	0.697	0.639	0.710
Boxes	**0.910**	0.677	0.634	0.701
Other	0.903	**0.732**	**0.650**	**0.720**

We took 64 images of similar scenes containing motors in boxes for evaluation. Since we want to evaluate whether the network can generalize, we also captured 24 images of configurations not present in the synthetic data. An example is depicted in Fig. 10c, where the motors are placed in a tray instead of a box.

We trained a Faster-RCNN network for 30 epochs, utilizing a dataset comprising 25,000 synthetic training images. The training was performed using SGD with an initial learning rate of 0.02, which we reduced by a factor of 0.1 after epochs 16 and 22. We chose the anime style as the intermediate representation since a one-channel representation has proven to be a good trade-off between inference speed and model accuracy in previous experiments. The detection results on the evaluation test set are compiled in Table 4. Interestingly, the detection accuracy between bin-picking and other cluttered scenes is very similar for a low IoU threshold. This indicates that the training synthetic training data is diverse enough to generalize the ROI classification to similar but unseen scenarios. The precision of box samples is lower compared to the rest of the validation data as the threshold increases. We attributed this to a higher probability of occlusion in those scenes, which makes it more challenging to locate object boundaries accurately.

One of the main advantages of our approach is that it does not require manual labeling, which is be time-consuming and costly. In traditional approaches using real-world images, image acquisition and labeling

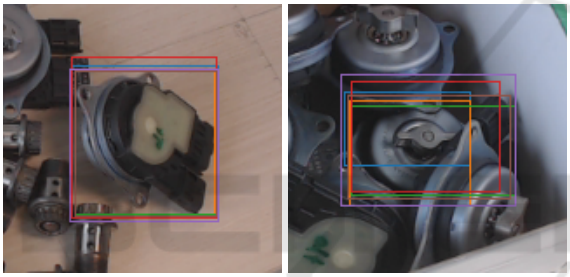


(a) Target Objects. (b) Training Sample. (c) Validation Sample.

Figure 10: Data used to train and evaluate the object detection use case.

Table 5: Comparison of labeling and data acquisition speeds.

	Box time	Acquisition time	Total
Synthetic	0.3s	9.0s	9.0s
Manual	8.4s	24.7s	57.2s



(a) Fully visible. (b) Occluded.

Figure 11: Two samples from our labeling study showing the annotations of all participants.

are separate steps that can add to the overall time and effort required. However, with our data-generation pipeline, labels (e.g., object bounding boxes) are generated simultaneously with the synthetic images. To quantify the reduction in manual labor provided by our approach, we measured the average time required to annotate a bounding box (box time) for both the training and evaluation datasets. We estimate the image acquisition time for the synthetic dataset by dividing the duration of synthetic data generation by the number of images in the training set. The image acquisition time is estimated using the duration of generating the synthetic dataset divided by the number of images in the training set. Note that this includes the time used to simulate the objects falling into the box. For the real-world dataset, we measure the time to capture the images and divide it by the dataset size.

A single user annotated the test dataset to ensure consistent label quality. However, the labeling speed is influenced by the user's experience, which introduces a bias in the timing statistics.

We conducted a small user study involving 10 participants to address this issue. Each participant was assigned a set of 10 images randomly selected from the test dataset and asked to annotate them. By comparing the results, we obtain a more comprehensive understanding of the labeling process and its associated time requirements. The findings from this study are presented in Table 5, revealing notable differences in image acquisition and labeling times between synthetic data generation and real-world scenarios.

Furthermore, we took advantage of this user study to investigate the impact of occlusion on the consistency of bounding box annotations across different users. For this purpose, two of the 10 presented images were identical for all participants. One depicts strong occlusion, and the other features a clear separation between instances. As hypothesized, we observed significant variations in annotated bounding boxes for heavily occluded motors, while instances with less occlusion generally exhibited more consistent labeling. Figure 11 provides visual examples for two instances from these images. For fully visible instances, annotation differences can be primarily attributed to sloppy labeling. However, achieving accurate annotations becomes exceedingly difficult when faced with occlusion and challenging lighting conditions.

In contrast, the labels of synthetic training data produced by our approach are inherently pixel-perfect, irrespective of the scene's complexity or level of occlusion.

5 CONCLUSION

This paper we presented a novel approach to synthetic image training. Instead of creating photo-realistic images or using neural networks that transform renderings into such images, we propose projecting synthetic and real-world data into a shared abstract domain. We demonstrated that line drawings are such

a domain, that is well suited for downstream tasks based on object geometry. Applying our approach to different tasks, we showed that the image-to-line transformation can be decoupled from these downstream tasks, and we presented various methods to facilitate the transformation. Our experiments showed that our method can be a drop-in to improve object detection quality, even using datasets with semi-realistic synthetic data. The intermediate line representation also enables novel augmentation methods, further improving network generalization to real-world data. Finally, we demonstrated how our approach could be used in a real-world use case by training a network to identify objects in a bin-picking scenario without any real training images.

Despite the success of our approach, there are areas for further exploration and optimization. The projection of images to their abstract representation is an additional step that requires computation time, and optimizing the runtime should be a focus in follow-up work. One promising idea is to use knowledge distillation with a student-teacher approach producing smaller image-to-image networks based on the presented ones. Additionally, we believe the downstream networks can be trimmed down since the abstract input data contains condensed, more meaningful data than pure color images.

ACKNOWLEDGEMENTS

This work has been supported by the Schaeffler Hub for Advanced Research at Friedrich-Alexander-Universität Erlangen-Nürnberg (SHARE at FAU).

REFERENCES

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Brachmann, E. (2020). 6D Object Pose Estimation using 3D Object Coordinates [Data].

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, pages 679–698.

Chan, C., Durand, F., and Isola, P. (2022). Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7915–7925.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Twenty-second international joint conference on artificial intelligence*. Citeseer.

Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. (2023). Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., and Santella, A. (2003). Suggestive contours for conveying shape. In *ACM SIGGRAPH 2003 Papers*, pages 848–855. ACM New York, NY, USA.

Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., and Katam, H. (2019). Blenderproc. *arXiv preprint arXiv:1911.01911*.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., and Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766.

Drost, B., Ulrich, M., Bergmann, P., Hartinger, P., and Steger, C. (2017). Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 2200–2208.

Foundation, B. (2022). Blender.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Goodman, N. (2022). Languages of art. In *Lexikon Schriften über Musik*, pages 293–376. Springer.

Harary, S., Schwartz, E., Arbelle, A., Staar, P., Abu-Hussein, S., Amrani, E., Herzig, R., Alfassy, A., Giryes, R., Kuehne, H., et al. (2022). Unsupervised domain generalization by learning a bridge across domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5280–5290.

Hertzmann, A. (2021a). The role of edges in line drawing perception. *Perception*, 50(3):266–275.

Hertzmann, A. (2021b). Why do line drawings work? a realism hypothesis. *Journal of Vision*, 21(9):2029–2029.

Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., and Zabulis, X. (2017). T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119.

Kennedy, J. M. and Ross, A. S. (1975). Outline picture perception by the songe of papua. *Perception*, 4(4):391–406.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Doll'a r, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

Peng, X., Usman, B., Kaushik, N., Wang, D., Hoffman, J., and Saenko, K. (2018). Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2021–2026.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). 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.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977.

Xie, S. and Tu, Z. (2015). Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403.

Zheng, Z., Wang, P., Ren, D., Liu, W., Ye, R., Hu, Q., and Zuo, W. (2021). Enhancing geometric factors in model learning and inference for object detection and instance segmentation. *IEEE Transactions on Cybernetics*, 52(8):8574–8586.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.