Classifying Cable Tendency with Semantic Segmentation by Utilizing Real and Simulated RGB Data

Pei-Chun Chien[1,2*], Powei Liao[2*], Eiji Fukuzawa[2], Jun Ohya[2]
[1]Yazaki Corporation, [2]Waseda University
peichun.chien@jp.yazaki.com, presley_liao@fuji.waseda.jp

Abstract

Cable tendency is the potential shape or characteristic that a cable may possess while being manipulated, of which some are considered erroneous and should be identified as a part of anomaly detection during an automatic manipulation. This research explores the ability of deep-learning models in learning the cable tendencies that, contrary to typical classification tasks of multi-object scenarios, is to differentiate the multiple states displayable by the same object – in this case, cables. By training multiple models with different combinations of self-collected real-world data and self-generated simulation data, a comparative study is carried out to compare the performance of each approach. In conclusion, the effectiveness of detecting three abnormal states and shapes of cables, and using simulation data is certificated in experiments.

1. Introduction

The development of robotic automation and its integration with conventional manufacturing have become unprecedentedly prevalent in many industries today. One common application is to utilize computer vision for inspection and detection, working in conjunction with robotic mechanisms to achieve an automated manufacturing line. For example, Khan et al. [10] explored a vision-guided inspection system for manufacturing parts, and Gregorio et al. [5] integrated a vision system into their wire-insertion manipulator. These works demonstrated the increasing reach of robot vision in manufacturing automation. In general, these vision systems utilize object recognition technology to learn relevant information necessary for the detection of their target objects.

Most robot vision systems focus on classifying different objects, from objects with distinct differences e.g., cars against pedestrians, to objects with semantic connotations, e.g., "tiles in a bathroom" which is more context-rich [12]. Such recognition and classification tasks are already widely seen in many applications today. However, not all scenarios benefit from this class diversity nor the ability to distinguish objects with significant differences. Instead, one may wish to distinguish a set of apparently identical object that possesses subtle differences in characteristics. An existing example is perhaps models that are capable of understanding car types, such as a truck, bus, or sedan [23]. While these classes can be broadly classified as "cars" or "automobiles", further training and appropriately diversified data have shown the ability of deep-learning models to extract minute features between them.

In this research, the specific scenario that fits the above description of context understanding is situated within a cable assembly production line. Typically, a human operator is required to install cables onto an assembly board while ensuring each cable is correctly routed as per design requirements. If this process is to be automated, an anomaly detection system capable of detecting faulty cable installation is first and foremost required to allow for subsequent rectifications. Such faults should be learned by having a model become tendency-aware and therefore capable of distinguishing potential anomalies.

A major challenge is the lack of such a dataset where tendencies are emphasized instead of object diversity. While datasets could be collected through manual recreation of target objects in the real world, it can be time-consuming. Recently, with advancements in the technology sector such as graphical computing unit (GPU) and simulation software, a common approach to the lack of data is to construct a simulation environment where data can be generated synthetically [17] [8]. Once constructed, the annotation is produced automatically at runtime, meaning the ground truth labeling work is handled instantly while being accurate down to the pixel-perfect level. The simulation can also be easily scaled up or run in parallel with multiple computers to generate a large quantity of data quickly, which is considered much more time efficient. Much suc-

[*]These authors contributed equally to this work

The work described by this paper is a collaborative research between Yazaki Corp. and Waseda University

cess has been observed in the field of self-driving cars [1], such as the CARLA simulator [6] and the NVIDIA DRIVE Sim by Omniverse [18]. These platforms provide an effective source of simulation data for deep-learning models to learn traffic-related information. Thus, given that the simulation data has drastically improved the recognition performance for self-driving cars, a similar effect can be anticipated when applied to cable tendency. Caveats exist, however, due to the domain difference between simulation and the real world, a 'reality gap' [4] exists that may hinder the learning ability of deep-learning models. This will be investigated and discussed throughout the paper.

Therefore, this paper aims to explore if deep-learning models can become tendency-aware through learning the subtle characteristic differences between numerous states and shapes of cables, assisted by simulated data. We collect real-world and synthetic RGB data of cables, where each image portrays normal and abnormal cases. The real-world data were collected by physically laying out cables onto an assembly board, whereas the simulation data were generated using the Blender 3D software [3]. In an effort to address the reality gap, we also applied domain randomization techniques. Through experimenting with different combinations of real and simulation data, multiple models were trained. The results show the inclusion of simulation data is capable of introducing slight improvements over models trained solely with real-world data, improving from a mean Intersection-over-Union (mIoU) of 75.54% up to 78.63% (trained with UPerNet [27] using *Swin_Transformer_Base* backbone [14]), and that by increasing simulation data proportion can improve a mixed-data model up to a point. A comparative analysis is carried out and the details are observed in this paper, laying a foundation for a generalizable approach to sim-to-real learning of object tendency that can be applied to robotic automation.

2. Related Work

In terms of cable detection, prior work has shown that through appropriate techniques, the detectors are able to realize the positions of thin and difficult-to-see cables [15,29]. Although the work by Li [13] and Pan et al. [19] achieved a promising 50.79% mean precision and 91.14% precision rate respectively, these systems do not differentiate cable characteristics nor define any fault. In the scope of this work, our models are to detect faulty cables instead of simply looking for the cable objects. A cable fault may be considered when it exhibits an overly loose, overly tensioned, or any other pre-defined erroneous shape, that appears to lie away from the main cable bundle due to its askew nature. Having to learn such information brings scene understanding into the equation. The semantic scene understanding, in contrast to object recognition, analyzes objects in context including their layout, spatial, function, and semantic relationship between all objects involved [7]. Studies have shown scene understanding to be achievable in numerous semantic segmentation tasks, a famous example is the brain tumor detector proposed by Ranjbarzadeh et al. [21] using Cascade Convolutional Neural Network. Their method is capable of distinguishing tumor cells from a series of MRI scans, despite the similarities in the majority of images. An improved version of the tumor detector proposed by Hatamizadeh et al. [9] further utilizes Swin Transformer which has shown competitive results in the BraTS 2021 challenge as one of the top performers. The Swin Transformer, published by the Microsoft team [14], introduces the novel **S**hifted **Win**dow (abbreviated as Swin) approach which has shown strong performance on COCO object detection and ADE20K semantic segmentation tasks. Due to the promising results, there has been an increasing presence of Swin Transformers in the field of segmentation tasks [16].

On the other hand, dataset availability remains an equally important element in any deep-learning task. Most existing open-source datasets place heavier emphasis on object diversity and not tendencies, therefore a new dataset is needed. While the real-world collection of data can be done through manual recreation, the process and subsequent labeling can take up a significant amount of time. For this, a common solution is to generate synthetic data. The idea of utilizing synthetic data often stems from the lack of existing data, as well as the simulation tool's ability to automatically generate annotations during the rendering process. Many studies have explored the usefulness of employing simulation data in deep-learning [17] [8], with most endorsing its ability to 1) cheaply produce a large quantity of data once the simulation environment is constructed, 2) automatically create pixel-level accurate labels, and 3) can be easily modified to better fit training requirements. However, caveats exist that the sole use of synthetic data introduces an adaptation gap between simulation and the real world – often referred to as the "reality gap" [24] [4]. With certain techniques such as improving photorealism, applying domain adaptation, or domain randomization, one may further improve the usability of generated synthetic data and help bridge said gap.

3. Materials & Methods

For this research, we begin by collecting a dataset of RGB cable images. Not only is this particular object class uncommon in off-the-shelf or open-source datasets, but the cable data should also be contextually and semantically meaningful where its tendencies are specifically labeled to show distinguishable features. These data could be gathered through capturing images of physically constructed scenes (hereinafter referred to as 'real data') or through the rendering of digitally generated scenes with a 3D simulation software (hereinafter referred to as ‘simulation data’).

Once the data are prepared, we set up four major models where each model is to be trained with a different combination of real and simulation data, allowing for a comparative study to be carried out. These include a real-only model (Model 1), a sim-only model (Model 2), a transfer-learning model (Model 3), and a mixed model (Model 4). For all models, Swin Transformer [14] is chosen and employed for training. While Swin Transformer provided backbones of several sizes [16] – Swin-Tiny, Swin-Small, Swin-Base, and Swin-Large – this paper explores Swin-Tiny and Swin-Base only. Each model is then tested against different sizes of simulation data and their results are compared and analyzed.

3.1. Cable Dataset

To produce a meaningful cable dataset, the images must portray tendencies of interest. Certain normal and abnormal cases were defined prior to the construction of cable layout scenes, and are used as the classes for detection. The classes, three error cases of which are shown in Fig. 1, are defined as follows:

- **Normal**: A bundle of cable consisting of five or more cables. This bundle follows a certain layout or route, which can be different for each scene construction.
- **Tensioned**: A cable that runs a shortcut at the inner-side of the normal cable bundle.
- **Loose**: A cable that protrudes unnecessarily at the outer-side of the normal cable bundle.
- **Twisted**: Two or more cables that are intertwined and twisted, showing periodically alternating colors.

Two types of data were produced, and are described below in detail:

3.1.1 Real Data

The real data are collected through physical reconstruction of the cable assembly scene. Using a decommissioned assembly board, cables are routed onto supporting components. Each scene contains at least one normal cable bundle and one erroneous cable that are laid according to the layout. The Torobo Eye SL40 camera by *Tokyo Robotics Inc.* is used to collect real data, as it provides highly accurate RGB, depth, and 3D point cloud information. An example of this camera can be seen in Fig. 2 (a). In the scope of this research, only the RGB information is used, and other data is saved for potential future work.

Two capturing methods were used in the collection of real data – mounted and unmounted. For the mounted approach, the camera is mounted onto a self-made aluminum frame as shown in Fig. 2 (b). The mount was adjusted to 50cm and 60cm from the ground. These distances were

In Fig. 1, Fig. 2, Fig. 3, Fig. 4, and Fig. 6, part of the images, except cables, may appear blurred due to confidentiality.

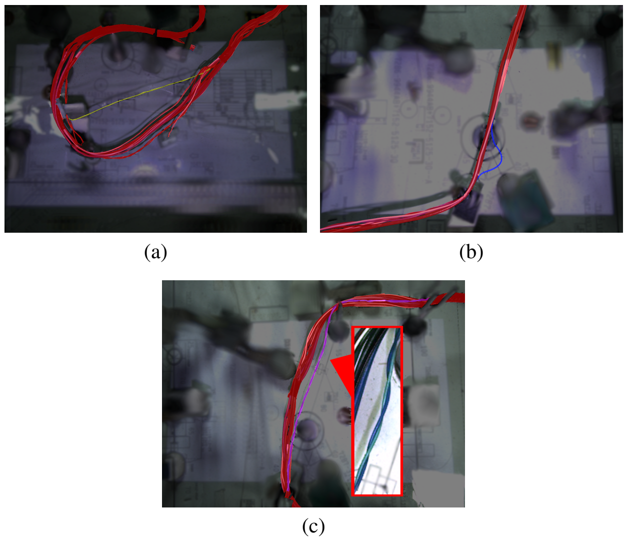
(a) (b) (c)

Figure 1. Examples of all three error cases. Each case contains a normal cable bundle as shown in red, whereas error cases are in non-red colors. Subfigure (a) shows the tensioned case, (b) = loose, and (c) = twisted.

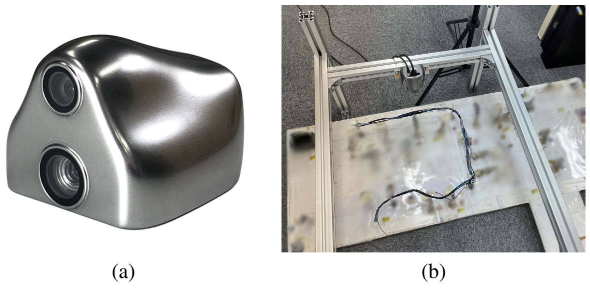
(a) (b)

Figure 2. (a) shows the employed SL40 camera, and (b) illustrates the scene setup used to collect real data.

chosen as the resultant images portray sufficient information about the scene, without experiencing any loss in image quality. The supporting components suspend the cables at around 3cm from the ground, meaning the cables were photographed at 47cm and 57cm away from the camera. Since the mount is attached to the railings of the aluminum frame, the mounted approach only allows a 90° angle during capture. For each distance, 10 images were collected. A series of fixed, patterned, and traceable real-world cable imagery data was created using these settings.

On the contrary, the unmounted approach intends to introduce diversity and randomness into the dataset. The camera was carried by a person and elevated at a similar distance above the ground at about 50 $\sim$ 60cm and tilted at 30° $\sim$ 60° angle. Since the camera was no longer restrained, the photographer circled the scene and captured 32 images per error case at varying distances and angles, totaling the real dataset to 156 real images. Image ground truths were then manually labeled using an annotation tool, *LabelMe* [26].

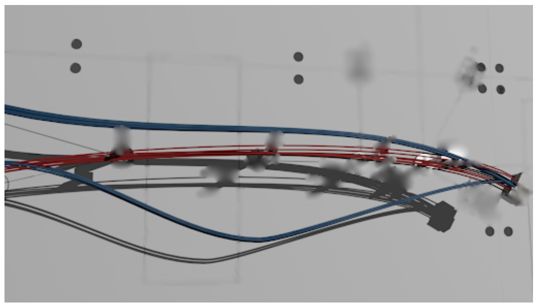

Figure 3. A rendered example of simulation data.

3.1.2 Simulation Data

The 3D environment was constructed in Blender 3D [3], an open-source software that handles the majority of computation involving manipulation, randomization, and rendering of the obtained simulation data. To create cables, the modified version of a community-made package – Cablerator [11] – was employed to simplify the cable creation process. An example of the simulated environment can be seen in Fig. 3.

The environment is built by defining pre-determined locations where scene components can spawn such as cable connectors and other static objects e.g., assembly board, layout visuals, and cable supporting components. A script is then used to handle all subsequent automation, which spawns several cables between random pairs of connectors. Upon execution of the script, the desired error type is to be passed in as a parameter, which limits the program to only spawn the given type of error. Of all the cables that spawn, each will have a chance of being either normal or the desired error type. The script is written in a way that guarantees at least one error and one normal cable to co-exist within a scene. Once the cables are in place, a physics simulation is processed to allow cables to free fall according to simulation gravity. This step is necessary as each error class behaves differently to physics simulation, and it ought to emphasize the characteristics of each class. Finally, the script proceeds to render three images each from a left-, centered-, or right-angled view. The annotation (ground truth) is automatically generated using Blender's *compositing* feature which, by assigning each object a 'pass index' according to its class and then converting the render to gray-scale, produces pixel-perfect annotation nearly instantly.

As mentioned in Section 2, in an effort to reduce the reality gap when crossing the real-world and simulation domain, certain aspects of the simulation were set to be random to achieve a level of domain randomization. To elaborate, Cable count (number of cables within a cable bundle), Cable material (color and texture), Cable attributes (tension, length, physics intensity etc.), Light source, and Camera angle were set to be random.

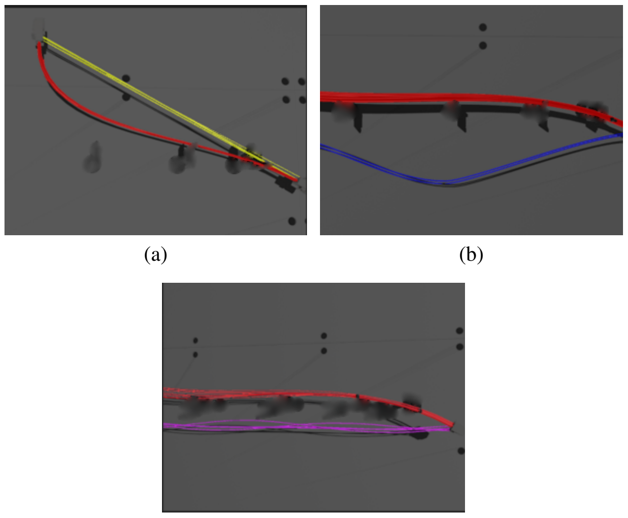

Figure 4. Examples of simulated error cable cases. Each case contains a normal cable bundle as shown in red, whereas error cases are in non-red colors. Subfigure (a) shows the tensioned case, (b) = loose, and (c) = twisted.

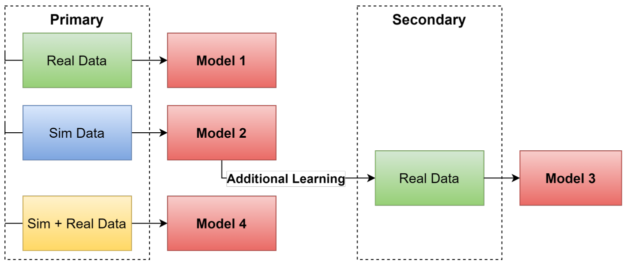

Figure 5. The overview of trainable model types.

As a result, more than 33,000 simulation images were generated, where Tensioned, Loose, and Twisted each make up about 11,000 images. An example of simulation images can be seen in Fig. 4.

3.2. Models for Classifying Cable Tendencies

As described at the beginning of this Section, four major model types utilizing different combinations of real and simulation data are to be trained. An overview of these models can be seen in Fig. 5.

The detail of each model is described as follows:

1. Model 1 (real-only) – for the purpose of comprehensiveness, this model is intended as a control group, which is trained solely using real data to seek the best performance achievable across types of deep-learning networks. Common segmentation networks and Swin-Transformer were selected for the experiment. The best-performing network will be used for subsequent training in Model 2, 3, and 4.

2. Model 2 (sim-only) – utilizes a varying number of solely simulation data as training input, which is then evaluated on real data.
3. Model 3 (transfer-learning) – extends on Model 2, but further feeds the model with available real data at a lower learning rate to accomplish a level of transfer learning. More details are described in Section 4.1.
4. Model 4 (mixed) – takes both real and simulated data as input from the beginning, but the process is repeated multiple times with various real-to-sim data ratios.

Each model is first trained with the Swin-Tiny backbone to observe a general trend between different data inclusions. Then, another iteration of training with the Swin-Base backbone is carried out for each model to affirm improvements. For evaluating the effect of different simulation data amounts, the Swin-Base backbone is used. In terms of complexity and model size, Swin-Tiny is about $0.25\times$ of Swin-Base, whose trainable parameters are about 29 million and 88 million respectively.

4. Experiments

4.1. Training

For all models except Model 3, Swin Transformer hyperparameters were kept default according to the source repository [16], this means employing the Adam optimizer, a learning rate of 6×10^{-5}, a batch size of 2, and using a fixed iteration of 160K for training. Upon training, image augmentation was done where images were randomly filled, scaled, rotated, sheared, and/or Gaussian blurred. The images are cropped to the size of 512×512 at random locations and will be randomly flipped. The learning rate will be a polynomial decay during the training. As for Model 3, due to its transfer-learning nature, a lower learning rate of 6×10^{-6} is used instead.

4.2. Dataset

As mentioned before, real and simulation datasets have been made. There are 156 images in the real dataset and 30k images in the simulation dataset. Different numbers of images and different ratios of real and simulation data have been used when training different models. For Model 1, a total of 111 real images are randomly selected as the training set and 45 real images are selected as the validation and test sets. For Model 2, 30k simulation images were used in training, where validation is 3,000 simulation images. The model is evaluated against real data with 45 real images. For Model 3, in terms of dataset amount, a similar approach to Model 1 is taken. For Model 4, a series of numbers of simulation data has been chosen to mix with all of the available 111 real images for training. The series is set as $75 \times 2^n (n = 0, 1, 2, ..., 8)$, up to 19,200 images. We begin with 75 simulation images as it is divisible by the three error classes available in our dataset. Besides the series, another training using 30,000 simulation images is also chosen as a point of interest. Model 4 again uses 45 real images as validation. Table 1 lists the number of images used for each model.

Table 1. An overview of used image amount per model.

Model	**Real Data**		**Sim Data**		**Test Data**	
	Train	*Val*	*Train*	*Val*	*S1*	*S2*
Model 1	111	45	*N/A*	*N/A*	45	45
Model 2	*N/A*	*N/A*	30k	3k	45	45
Model 3	111	45	30k	*N/A*	45	45
Model 4	111	45	75~30k	*N/A*	45	45

To verify if these models can be used in different environments, in addition to the original scenario where the training set is built, two different scenarios have also been created. In scenario 1 (*S1* in Table 1), a self-made assembly board was employed. To exaggerate differences, scenario 1 adopted a yellow board instead of white, and utilized self-made cable-supporting components. The layout was imitated using black tapes, which appear texturally identical to those wrapped around certain parts of cables. This decision was to increase the difficulty and to introduce background diversity. In scenario 2 (*S2* in Table 1), a similar board to that of the original was used. However, the background layout is distinct from the original board, and the cables were laid in a more crowded and complex nature.

5. Results

Model 1 is trained and evaluated against real data. As mentioned earlier, several networks were tested for a comprehensive comparison. The results can be seen in Table 2, where Swin-Base + UPerNet achieved the highest performing mIoU of 75.54% from the original test scene. The Swin-Base + UPerNet combination is therefore used for the training of Model 2, 3, and 4. The reason for its high performance is likely contributed by the shifted window self-attention technique employed in Swin Transformer. The self-attention technique attempts to summarize input information globally to learn to focus on important regions [25] [20]. Swin Transformer computes the self-attention locally through window partitioning while applying stitching at necessary window segments to maintain cross-window connections without much complexity overheads. Such a technique is likely the reason for its higher performance as opposed to other CNN networks.

A series of inference results of Model 1 can be seen in row 3 of Fig. 6. The results show that Model 1 is capable of distinguishing classes correctly to an extent, but misclassification of error cases and false-identification of normal cables occur quite frequently, especially in scenario 1 where

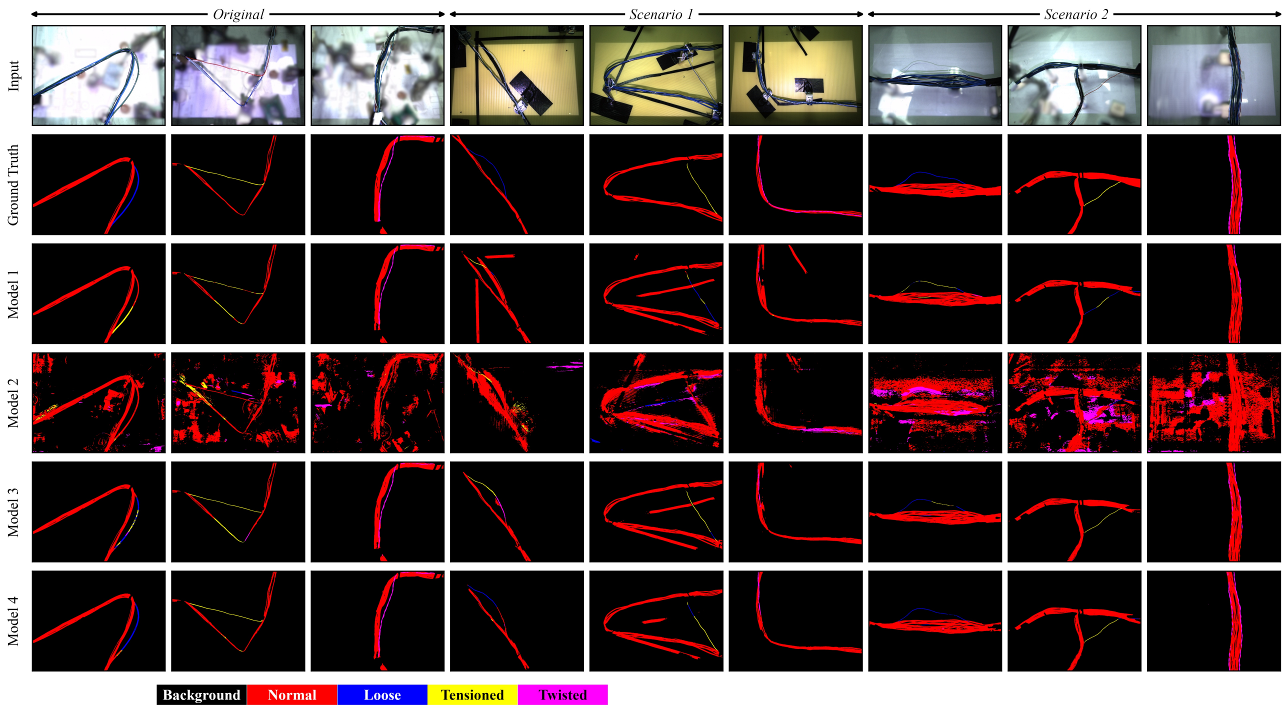

Figure 6. The inference results of each model against each class.

Table 2. Comparison of segmentation networks.

Real-data-only comparison		
Backbone	**Method**	**mIoU (%)**
VGG-16	U-Net [22]	34.72
VGG-16	PSPNet [28]	27.20
VGG-16	SegNet [2]	43.92
ResNet50	U-Net	58.16
ResNet50	PSPNet	29.45
ResNet50	SegNet	42.08
Swin-Tiny	UPerNet [27]	73.62
Swin-Base	UPerNet	**75.54**

the background layout possesses a similar appearance to cables. Detailed class-wise IoUs can be seen in Table 3. The table also shows that in different test scenes – scenario 1 and 2 – Model 1 experienced a drop in performance as compared to the original test scene, where the twisted class experienced the largest drop from 81.01% to 20.76% IoU. While this is expected to a certain degree as these scenes were specifically made with increased difficulty, an IoU difference of 60% indicates that the model struggled with generalizing the twisted cables.

Model 2, trained solely with simulation data, was evaluated against real data. Employing the Swin-Base backbone, Model 2 achieved 21.57% mIoU against the original test scene, as shown in Table 3. Compared to Model 1's 75.54% mIoU, this significant drop in performance was expected due to the reality gap between the simulation and the real world which, despite efforts taken to mitigate the differences, the cross-domain discrepancy remains a substantial obstacle. Further evaluation against scenario 1 and 2, Model 2 obtained 25.35% and 18.17% mIoU respectively. Table 3 also details class-wise IoUs that, while every class suffered a similarly low performance, the loose class was observed as the most degraded with no correct identification at 0% IoU. The inference results of Model 2 can be seen in row 4 of Fig. 6. The results show that while most of the normal cables can be identified, Model 2 lacks the ability to meaningfully distinguish error classes and is prone to mis-classify background noises.

Model 3 achieved a mIoU of 76.67% when evaluated against the original test scene. It is observed that Model 3 performs superiorly to Model 1, which is likely due to the additional simulation information learned during the Model 2 phase, and having it rectified with real data allows it to surpass Model 1 slightly. When evaluated against scenario 1 and 2, Model 3 achieved 56.88% and 58.94% mIoU respectively. As shown in row 5 of Fig. 6, the model was able to correctly classify more parts of the cables, but some common errors remain noticeable to that of Model 1's results. However, the false-identification of background layouts in scenario 1 has significantly reduced, which helped improve the general mIoU. The detailed class-wise IoU is

Table 3. Performance evaluation results of Model 1 to 4 trained with **Swin Transformer + UPerNet**.

Model	Backbone	mIoU	background	normal	loose	tensioned	twisted	Test Scene	Real	Sim
Model 1	Swin-Base	75.54	99.51	84.93	47.73	64.54	81.01	original	111	*N/A*
Model 2	Swin-Base	21.57	85.56	17.22	0	2.91	2.15	original	*N/A*	30k
Model 3	Swin-Base	76.67	99.51	85.13	56	65.49	77.25	original	111	30k
Model 4	Swin-Base	**78.63**	99.52	85.52	55.63	73.72	78.78	original	111	2.4k
Model 1	Swin-Base	53.42	96.96	47.58	46.55	55.82	20.18	scenario 1	111	*N/A*
Model 2	Swin-Base	25.35	90.43	21.47	0	11.42	3.41	scenario 1	*N/A*	30k
Model 3	Swin-Base	56.88	98.51	63.41	45.17	54.67	22.63	scenario 1	111	30k
Model 4	Swin-Base	**59.43**	98.67	66.12	54.33	58.21	19.84	scenario 1	111	300
Model 1	Swin-Base	59.74	99.25	85.5	45.95	47.26	20.76	scenario 2	111	*N/A*
Model 2	Swin-Base	18.17	71.59	15.22	0	3.37	0.67	scenario 2	*N/A*	30k
Model 3	Swin-Base	58.94	99.29	86.2	44.08	51	14.11	scenario 2	111	30k
Model 4	Swin-Base	**62.62**	99.25	85.71	51.62	49.32	27.19	scenario 2	111	300

Table 4. *Expanded Model 4* utilizing partial data from scenario 1 and 2.

Backbone	mIoU	background	normal	loose	tensioned	twisted	Test Scene	Real	Sim
Swin-Base	77.55	99.54	85.89	54.44	68.16	79.72	original	141	300
Swin-Base	74.05	99.50	83.87	63.87	75.52	50.48	scenario 1	141	300
Swin-Base	68.66	99.51	90.29	47.68	59.79	46.04	scenario 2	141	300

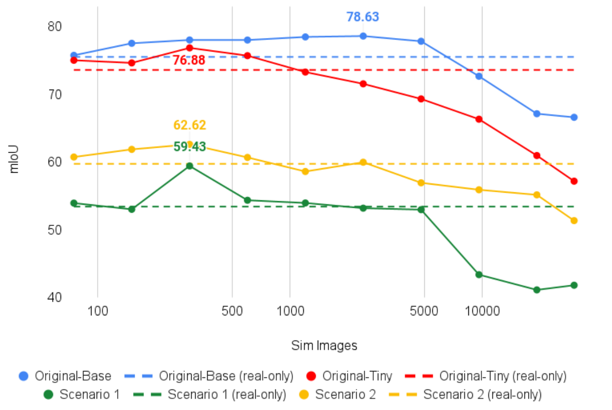

Figure 7. Performance trend of Model 4 compared against Model 1. The maximum of each series is labeled on the graph.

again shown in Table 3, which shows that Model 3 is also unable to generalize the twisted class very well. Model 4 was trained with various real and simulation data proportions and was tested against real data. With the Swin-Base backbone, the optimal performance is found achieved at 78.63% mIoU using 2,400 simulation images when evaluated on the original scene. When repeated for evaluation against scenario 1 and scenario 2, the optimal results were obtained by using 300 simulation images, at 59.43% and 62.62% mIoUs respectively.

Figure 7 shows the plotted mIoU results of Model 4 at different simulation data inclusion. In this plot, solid lines indicate the Model 4 mIoUs at different ratios. The dashed lines of relevant colors indicate the mIoU of the real-only model under the same scenario setting. These together show the comparison between the real-only model and Model 4's fluctuation at different simulation data inclusion.

While the experiment mainly focused on employing the Swin-Base backbone, a Swin-Tiny Model 4 was also trained and illustrated in red in Fig. 7. The idea is to show that, as demonstrated in Table 2, the superior performance of Swin-Base over Swin-Tiny backbone is also maintained in Model 4. The inference example is shown in row 6 of Fig. 6. The images show Model 4's improved capability in identifying cables and correctly classifying most classes. Similarly, the class-wise IoU is detailed in Table 3. When evaluated against the original scene, most classes were able to obtain above 70% IoU except for the loose class at only 55.63%. In cases of scenario 1 and 2, Model 4 experienced performance degradation where the twisted class again sustained the most noticeable reduction, in which the model acquired 19.84% and 27.19% IoU respectively.

From the results, it appears that models using simulation data as an additive input (e.g., Model 3 and 4) introduced leveraging improvement over those trained solely with one type of data (e.g., Model 1 and 2). It is observed that the twisted class appears not well-generalized, which resulted in a noticeable degradation in mIoU, especially in non-original test scenes. To allow better generalizability, an approach is to partition a small amount of scenario 1 and 2 images into the training set. An *Expanded Model 4* was created where 5 images of each error class (loose, tensioned, and twisted) of both scenario 1 and 2 were added into the training pool, totaling to an additional 30 images. The simulation image inclusion is maintained at 300 images as Model 4 with this amount produced the best result when evaluated against scenario 1 and 2. The result of *Expanded Model 4* can be seen in Table 4. The expanded version shows the model mIoU significantly better maintained when tested on scenario 1 and 2. The class-wise IoU also shows the twisted class experiences a much less performance drop, situating at about 50% for both non-original scenes.

6. Discussion

The primary goal of this research is to gain an understanding of what type or combinations of data can better assist the learning of cable tendency. The task focuses not on object diversity nor the ability to detect object presence but instead aims to extract subtle characteristics between the possible states of an apparently identical object – in this case, cables. By constructing a new dataset that emphasizes these tendencies, we have observed that deep-learning models are capable of learning such subtlety. The use of simulation data was hypothesized as an additive factor that should introduce improvement into deep-learning models.

We observed that using solely simulation data (Model 2), the impact of the reality gap drastically reduces the performance despite applying domain randomization. In an attempt to amend this gap, Model 3 and 4 were trained to observe the differences. By feeding real data into the system afterward, Model 3 showed its ability to rectify its sim-only model effectively. The results of Model 4 showed that the use of simulation images is capable of providing improvements up to an extent. From Table 3, Model 4 appears to arrive at a peak at 78% from 300 to 2,400 simulation images. Treating 300 as the beginning of this peaked learning, as illustrated by Fig. 7, we observe the simulation data inclusion to be beneficial at about 1:3 real-to-sim data ratio (111:300). Using more simulation data may slightly improve the mIoU, but the improvement is limited and negligible. Going beyond this ratio, the performance started to drop as the model again gravitated toward the simulation domain and fails to effectively extract useful information. At this point of our experiment, we have demonstrated the appropriate proportion of real and simulation data can in fact positively affect cable tendency learning.

On the other hand, evaluations on non-ordinary test scenes showed a rather impactful performance degradation. Using Model 4 as the main comparator, the original, scenario 1, and scenario 2 test scenes each obtained 78.63%, 59.43%, and 62.62% respectively. The class-wise IoUs suggest some classes, such as the twisted class, were not well-generalized and performed inferiorly on un-trained scenes. An expanded Model 4 was therefore constructed and tested which, by merely including 5 images of each class from the new scenes, the model was able to significantly improve the IoU of nearly all classes. Again using the twisted class as an example, Table 4 shows that while the original Model 4 only obtained a class IoU of 19.84% and 27.19% in scenario 1 and 2 respectively, the expanded Model 4 obtained 50.48% and 46.04% IoU respectively. The overall mIoU performance of expanded Model 4 also greatly improved, going from 59.43% $\rightarrow$ 74.05% mIoU in scenario 1 and 62.62% $\rightarrow$ 68.66% mIoU in scenario 2. It is therefore recommended that when testing on a new scenario, due to differences in layouts and scene objects, a very small amount of annotated data from this new scenario should be fed into the model as a part of the training set to further broaden the generalizability of the model.

7. Conclusion

This work aimed to explore the ability of deep-learning models in understanding cable tendencies, and by employing what type and combination of data can best assist such learning. Our work has shown the effectiveness of simulation data in assisting deep-learning models in becoming tendency-aware. Using real-only data, Model 1 shows promising capability in learning tendencies but remains improvable. Conversely, Model 2 using simulation-only data has limited performance when evaluated against real data due to the domain difference or 'reality gap'. By practicing transfer learning in Model 3, the model is capable of rectifying its predecessor (Model 2) into one that understands real data, obtaining a much-improved result from that of Model 2. With Model 4, we demonstrated simulation data effectively improves the performance at the appropriate ratio (1:3) as shown in Fig. 7. Testing on new scenarios also shows a drop in performance, but is rectifiable by introducing a very slight partition of their data into training. In terms of simulation data, the success is attributed to the completeness of the simulation environment design – by introducing more randomness into the simulation environment, the generated output should have a further domain reach and may help with bridging the reality gap. In conclusion, our experiment suggests tendency learning is possible and it is recommended to employ the mixed-data approach similar to that of Model 4 as it can effectively realize tendency detection without needing an overly large amount of simulation images. Model 4 also performed the most optimally compared to others, despite the improvement is slight. In this work, there are several extensions that we wish we had more time to explore:

1. Implementing an alternative evaluation method utilizing weight function in additional to pixel-wise mIoU.
2. Expanding the dataset both in the real world and simulation environment.

References

[1] David Acuna, Jonah Philion, and Sanja Fidler. Towards optimal strategies for training self-driving perception models in simulation. *CoRR*, abs/2111.07971:1–19, 2021. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561:1–14, 2015. 6

[3] Blender Foundation. Blender - a 3d modelling and rendering package, 2022. 2, 4

[4] Jan Blumenkamp, Andreas Baude, and Tim Laue. Closing the reality gap with unsupervised sim-to-real image translation for semantic segmentation in robot soccer. *CoRR*, abs/1911.01529:1–12, 2019. 2

[5] Daniele De Gregorio, Riccardo Zanella, Gianluca Palli, Salvatore Pirozzi, and Claudio Melchiorri. Integration of robotic vision and tactile sensing for wire-terminal insertion tasks. *IEEE Transactions on Automation Science and Engineering*, 16(2):585–598, 2019. 1

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 2

[7] Max Planck Institute for Intelligent Systems. Semantic scene understanding. 2

[8] Pierre Gutierrez, Maria Luschkova, Antoine Cordier, Mustafa Shukor, Mona Schappert, and Tim Dahmen. Synthetic training data generation for deep learning based quality inspection. *CoRR*, abs/2104.02980:1–8, 2021. 1, 2

[9] Ali Hatamizadeh, Vishwesh Nath, Yucheng Tang, Dong Yang, Holger Roth, and Daguang Xu. Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images. *BrainLes 2021*, abs/2201.01266:272–284, 2022. 2

[10] Aamir Khan, Carmelo Mineo, Gordon Dobie, Charles Macleod, and Gareth Pierce. Vision guided robotic inspection for parts in manufacturing and remanufacturing industry. *Journal of Remanufacturing*, 11(1):49–70, Apr 2021. 1

[11] Sergey Kritskiy. Cablerator, 2021. 4

[12] Tim Lauer, Filipp Schmidt, and Melissa L.-H. Võ. The role of contextual materials in object recognition. *Scientific Reports*, 11(1):21988, Nov 2021. 1

[13] Yingyu Li. Object detection and instance segmentation of cables, 2019. 2

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030:1–14, 2021. 2, 3

[15] Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3487–3494, 2017. 2

[16] Microsoft. Swin-transformer, 2022. 2, 3, 5

[17] Sergey I. Nikolenko. Synthetic data for deep learning. *CoRR*, abs/1909.11512:1–156, 2019. 1, 2

[18] NVIDIA. Nvidia drive sim powered by omniverse. 2

[19] Chaofeng Pan, Xianbin Cao, and Dapeng Wu. Power line detection via background noise removal. In *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 871–875, 2016. 2

[20] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909:1–15, 2019. 5

[21] Ramin Ranjbarzadeh, Abbas Bagherian Kasgari, Saeid Jafarzadeh Ghoushchi, Shokofeh Anari, Maryam Naseri, and Malika Bendechache. Brain tumor segmentation based on deep learning and an attention mechanism using MRI multi-modalities brain images. *Sci Rep*, 11(1):10930, May 2021. 2

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597:1–8, 2015. 6

[23] Li Suhao, Lin Jinzhao, Li Guoquan, Bai Tong, Wang Huiqian, and Pang Yu. Vehicle type detection based on deep learning in traffic scene. *Procedia Computer Science*, 131:564–572, 2018. Recent Advancement in Information and Communication Technology. 1

[24] Joshua 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. *CoRR*, abs/1703.06907:1–8, 2017. 2

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762:1–15, 2017. 5

[26] Kentaro Wada. Labelme: Image Polygonal Annotation with Python. GitHub. 3

[27] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. *CoRR*, abs/1807.10221:1–18, 2018. 2, 6

[28] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105:1–11, 2016. 6

[29] Chen Zhou, Jiaolong Yang, Chunshui Zhao, and Gang Hua. Fast, accurate thin-structure obstacle detection for autonomous mobile robots. *CoRR*, abs/1708.04006:1–10, 2017. 2