remote sensing

MDPI

Article

Detecting Object-Level Scene Changes in Images with Viewpoint Differences Using Graph Matching

Kento Doi [1,2,*], Ryuhei Hamaguchi [2], Yusuke Iwasawa [1], Masaki Onishi [2], Yutaka Matsuo [1] and Ken Sakurada [2]

[1] School of Engineering, The University of Tokyo, Tokyo 113-0033, Japan
[2] Artificial Intelligence Research Center (AIRC), The National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan
* Correspondence: doi.kento@weblab.t.u-tokyo.ac.jp; Tel.: +81-03-5841-7718

Abstract: We developed a robust *object-level* change detection method that could capture distinct scene changes in an image pair with viewpoint differences. To achieve this, we designed a network that could detect object-level changes in an image pair. In contrast to previous studies, we considered the change detection task as a graph matching problem for two object graphs that were extracted from each image. By virtue of this, the proposed network more robustly detected object-level changes with viewpoint differences than existing pixel-level approaches. In addition, the network did not require pixel-level change annotations, which have been required in previous studies. Specifically, the proposed network extracted the objects in each image using an object detection module and then constructed correspondences between the objects using an object matching module. Finally, the network detected objects that appeared or disappeared in a scene using the correspondences that were obtained between the objects. To verify the effectiveness of the proposed network, we created a synthetic dataset of images that contained object-level changes. In experiments on the created dataset, the proposed method improved the F_1 score of conventional methods by more than 40%. Our synthetic dataset will be available publicly online.

Keywords: change detection; object detection; graph matching

Citation: Doi, K.; Hamaguchi, R.; Iwasawa, Y.; Onishi, M.; Matsuo, Y.; Sakurada, K. Detecting Object-Level Scene Changes in Images with Viewpoint Differences Using Graph Matching. *Remote Sens.* **2022**, *14*, 4225. https://doi.org/10.3390/rs14174225

Academic Editor: Pedro Melo-Pinto

Received: 8 July 2022
Accepted: 18 August 2022
Published: 27 August 2022

1. Introduction

The rapid detection of environmental changes that are caused by artificial factors or natural phenomena is crucial in remote sensing and robotics. For example, an instantaneous understanding of geometric changes within the environment and keeping environmental maps up to date are crucial for ensuring navigational accuracy in self-driving vehicles and indoor service robots. Change detection [1,2] is a technology that automatically detects environmental changes in data from sensors and has been studied using satellite [3–8], aerial [9–11] and street view images [12–21]. Among these, change detection using street view images can capture detailed changes that cannot be captured in satellite or aerial images; therefore, it is expected to be employed in various applications, such as the updating of maps for self-driving vehicles.

The most typical change detection method for street view images estimates changes pixel by pixel. Specifically, it computes the probability of the presence or absence of each pixel in each image. For example, the methods that were proposed by Taneja et al. [12] and Palazzolo et al. [14] estimated the changes in urban areas at the pixel level based on image differences between aligned images using dense 3D maps. Recently, methods that are based on convolutional neural networks (CNNs) have also been developed [13,17,18,20,21]. They are inspired by the use of CNNs for semantic segmentation, which is achieved by training models using large sets of labeled data. For example, Alcantarilla et al. [13] and Heo et al. [20] proposed a method for change detection using a network in which all of the layers were convolutional layers.

Although many change detection methods have been studied at the pixel level, the ability to handle changes at the object level exceeds the range of their application. For example, some 3D maps that are used for navigation in self-driving vehicles and robots contain information about geographic objects at the object level. Updating such maps by removing existing objects or adding new objects requires the detection of changes in the units of the objects. The detection of changes at the object level is also useful for counting the number of changed objects. Unfortunately, even though object-level change detection is a very promising technique, it has rarely been explored in studies on street view images.

Although there has been limited research on object-level change detection using street view data, several studies have been conducted using satellite images [7,22–24]. In object-level change detection, the key is how to extract object regions and over recent years, methods that use deep learning have been proposed for this task [22–24]. For example, Zhang et al. [24] proposed a method to detect changes in bounding boxes using an object detector that was implemented with a convolutional neural network. With the development of these techniques, it has become possible to accurately detect object regions, as long as sufficient annotated data exist. However, applying the model for satellite images directly to street view data is difficult since viewpoint differences are likely to occur between images. Due to differences in driving paths and shutter timing, two images that are captured at different times by a camera that is mounted on a vehicle may contain viewpoint differences. Furthermore, in contrast to orthorectified satellite images, street view images cannot be aligned using parallel shifts. When dense 3D models exist, it is possible to align images accurately, as demonstrated in previous studies [12,13]; however, it is not feasible to maintain high-quality 3D data at the city/national level because of the associated costs. Typically, change detection methods assume image alignment and their accuracy becomes an issue when there are viewpoint differences between the images. Recently, change detection networks that are robust to viewpoint differences have been proposed. However, the relationship between disparity scale and change detection accuracy has not been comprehensively investigated and remains an ongoing challenge.

To this end, we propose a robust method for object-level change detection in images with viewpoint differences that utilizes object detection and a graph matching network (Figure 1). In contrast to existing methods, our method does not depend on pixel-level predictions. Instead, we formulated the problem as a graph matching problem between two object graphs that are extracted from each image, which does not require pixel-wise image alignment and is therefore robust to viewpoint differences. Specifically, the proposed network detects object-level changes in three steps. Firstly, the object detection module (e.g., Faster R-CNN [25]) detects the regions and visual features of the target objects in each image. Secondly, the graph matching module constructs the correspondences between the objects by considering the similarities and relationships between them. Finally, the regions and classes of objects with no matching pairs are categorized as appearing/disappearing objects in each scene. We noted that the object detection module in the proposed network could also be replaced with an instance segmentation network to predict changes using instance masks instead of bounding boxes. Moreover, the entire network can be trained using the supervised loss of object detection and graph matching; thus, it does not require annotations for pixel-level change masks, which are labor-intensive and time-consuming.

In addition to the methodological proposal, we created a synthesized object-level change detection (SOCD) dataset, which was the first dataset that could be used to evaluate object-level change detection. The SOCD dataset comprised approximately 15,000 perspective image pairs that were rendered by a CARLA simulator [26]. Furthermore, to investigate the relationships between the scale of the viewpoint differences and the accuracy of the change detection method, we created image pairs with four separate scales of viewpoint differences for the same scene. In our experiments on the SOCD datasets, the proposed method improved the F_1 score of existing methods by more than 40%.

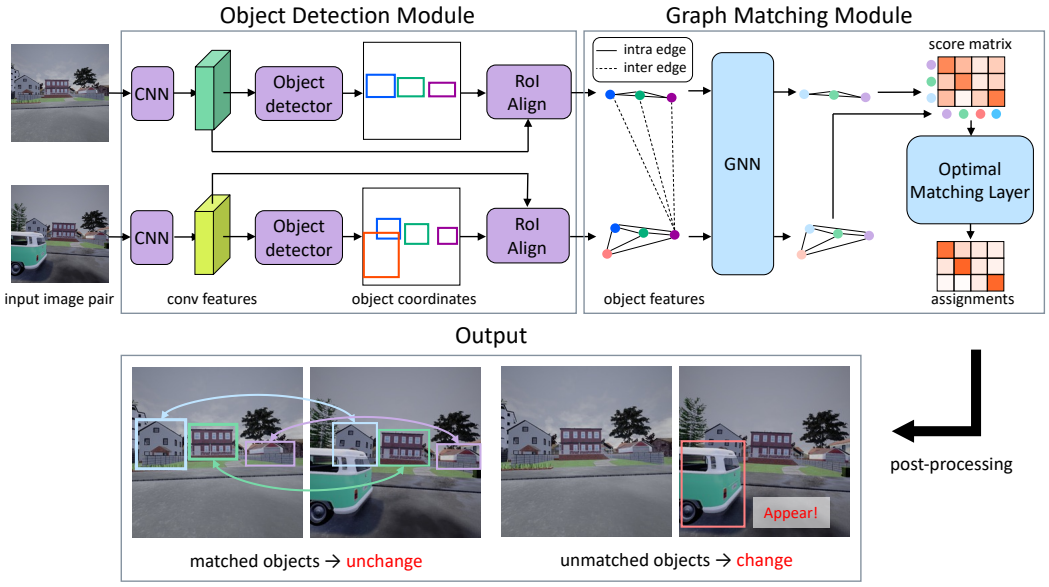

Figure 1. An overview of our alignment-free object-level change detection network. The network includes an object detection module (as discussed in Section 3.2) and a graph matching module (as discussed in Section 3.3). The object detection module detects object regions and creates object features. The graph matching module accepts the object features as input and constructs assignments using a graph neural network and an optimal matching layer. Finally, the regions of unmatched objects are categorized as object-level changes.

The main contributions of this paper are as follows:

- The proposal of as object-level change detection network that is robust to viewpoint differences, which can be trained using bounding boxes and the correspondences between them;
- The development of an SOCD dataset to allow for the study of the performance of object-level change detection methods at different scales of viewpoint differences (the SOCD dataset will be publicly available);
- Experiments on the developed dataset that confirmed that the proposed method outperformed the baseline method in terms of the accuracy of object-level change detection.

2. Related Work

2.1. Image Change Detection

Image change detection [2] is the task of detecting scene changes within an image. The basic approach is to estimate the probability of change for each pixel by comparing images that were taken at different times from the same viewpoint and various methods have previously been studied. An important recent trend is the use of deep learning for more accurate change estimation. For example, Sakurada et al. [27] proposed a method for predicting change regions based on the differences in visual features that were extracted by a CNN. Alcantarilla et al. [13] trained a deconvolutional network to detect structural changes in street view images. Guo et al. [28] applied contrastive loss, which encouraged CNNs to learn more discriminative features to improve their robustness to noise, such as illumination or camera poses. Lei et al. [21] proposed a deep CNN that utilized the adaptive fusion of image features.

One important factor to deal with is viewpoint differences between images. For example, when capturing images with cameras mounted on automobiles or mobile robots, most of the conditions are variable, such as camera trajectory, mounting position and shutter timing; therefore, capturing images with exactly the same camera pose is challenging. One of the most straightforward approaches to address this is to align the images using a dense 3D model. For example, Taneja et al. [12] and Palazzolo et al. [14] proposed change detection methods for 3D map updates that leveraged existing 3D maps to perform this

alignment. Alcantarilla et al. [13] utilized geometric information that was reconstructed from image sequences for image alignment to detect changes using a deconvolutional network. However, the costs of the data collection and computation that are required to obtain exact 3D models are extremely high. By contrast, several approaches do not depend on dense 3D models to detect changes. For example, Sakurada et al. [18] introduced a robust network architecture that used correlation layers. Furukawa et al. [17] proposed a method for learning change detection and image alignment without using a hand-labeled dataset. These two methods achieved robust change detection for images with viewpoint differences. However, the method that was proposed by Sakurada et al. [18] had a problem in that the correlation layers could not deal with large viewpoint differences and the method that was proposed by Furukawa et al. [17] could only learn to predict binary change masks.

Another important issue is the detection of changes at the object level. For example, Hazel et al. [7] proposed a method for detecting changes in spectral imagery at the object level by combining multiple steps, such as object region/feature extraction, object association and change detection. This method, which performed object extraction and mapping sequentially, is analogous to our proposed method. For object-level change detection, it is important to accurately extract object regions, as well as correctly determine changes. Recently, change detection methods that utilize deep learning-based object detection networks have been proposed, such as Faster R-CNN [25] and Mask R-CNN [29] [22–24]. However, an unsolved problem is how to match objects when there are viewpoint differences between images. In particular, there has not yet been any research on object-level change detection for street view images with viewpoint differences, which was the target of this study.

2.2. Graph Matching

Graph matching [30–34] is a method for establishing node correspondence between two graphs. Recently, deep learning-based methods have enabled end-to-end learning and have significantly improved performance [32–34]. Out of the numerous existing methods, we mainly focused on differentiable (or deep learning-based) graph matching methods that could be incorporated into our end-to-end change detection network.

Deep learning-based graph matching methods can solve these problems in three steps: (i) the extraction of node (and edge) features; (ii) the prediction of score matrices; and (iii) the assignment of nodes from the score matrices. For example, Zanfir and Sminchisecu [32] developed the first deep learning-based graph matching method by extending each computation for factorized graph matching to differentiable graph matching. Wang et al. [33] and Sarlin et al. [34] proposed methods that utilized GNNs [35,36] to predict the score matrices in order to consider the higher structural information of the graphs. The method that is the most related to our approach is the SuperGlue network that was proposed by Sarlin et al., which is a graph matching network with an attentional GNN and an optimal matching layer that was based on the Sinkhorn algorithm [37,38]. In this study, we used the SuperGlue network as the graph matching module in our change detection network.

3. Methods

Our main challenge was to develop an architecture that could detect *object-level* changes, bounding boxes or instance masks to capture changes in objects from *unaligned* image pairs, such as crowdsourced images. As shown in Figure 1, we handled the challenge by (i) extracting object regions and features from each image using an object detection module and (ii) matching the objects to detect any changes using a graph matching module that was robust to viewpoint differences. Section 3.1 describes the formulation of the proposed approach. Sections 3.2 and 3.3 describe the details of the object detection and graph matching modules, respectively. Finally, Section 3.4 describes the loss function of the proposed network.

3.1. Formulation

In this subsection, we discuss the object-level change detection problem. First, we considered the images $\boldsymbol{I}_1 \in \mathbb{R}^{3\times W\times H}$ and $\boldsymbol{I_2} \in \mathbb{R}^{3\times W\times H}$, which captured the same scene at times t_1 and t_2 ($t_1 < t_2$), where W, H are the width and height of the images, respectively. Then, the change detection model received $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ and returned the bounding boxes $\boldsymbol{B}_1 \in \mathbb{R}^{N_1\times 4}$ and $\boldsymbol{B}_2 \in \mathbb{R}^{N_2\times 4}$ or the instance masks $\boldsymbol{M}_1 \in \mathbb{R}^{N_1\times W\times H}$ and $\boldsymbol{M}_2 \in \mathbb{R}^{N_2\times W'\times H'}$, which localized the objects in $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, where $N_1, N_2 \in \mathbb{N}$ denote the number of changed objects in $\boldsymbol{I}_1, \boldsymbol{I}_2$, respectively, and W', H' denote the width and height of each mask, respectively. Whether the outputs were bounding boxes or instance masks depended on the structure of the object detection module, as described in Section 3.2.

We solved the problem of object-level change detection as follows. Firstly, the object detection network accepted $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ as inputs and extracted M and N objects from them, respectively. Next, the graph matching module predicted the correspondences between these objects as an assignment matrix $\boldsymbol{P} \in \{0,1\}^{M\times N}$, where $\boldsymbol{P}_{i,j} = 1$ when the i-th object in image $\boldsymbol{I}_1$ and the j-th object in image $\boldsymbol{I}_2$ matched and $\boldsymbol{P}_{i,j} = 0$ when they did not match. Each object had a maximum of one matching pair. Because a scene could contain changes, there could be objects with no matching pair. Hence, the assignment matrix satisfied the following conditions:

$$\boldsymbol{P}\mathbf{1}_N \leq \mathbf{1}_M, \quad \boldsymbol{P}^\top \mathbf{1}_M \leq \mathbf{1}_N, \tag{1}$$

where $\mathbf{1}_M$ is an all-ones vector of length M. Finally, the changes were determined for each object using the predicted assignment matrix $\hat{\boldsymbol{P}}$. In this study, we considered two types of changes: *appearance* and *disappearance*. The jth object in $\boldsymbol{I}_2$ was determined to appear when $\sum_{k=1}^{M} \hat{\boldsymbol{P}}_{k,j} = 0$. Similarly, the ith object in $\boldsymbol{I}_1$ was determined to disappear when $\sum_{l=1}^{N} \hat{\boldsymbol{P}}_{i,l} = 0$. We neglected other types of changes (e.g., the same object with a different texture) for future work.

3.2. Object Detection Module

The object detection module extracted objects in the scenes from the pair of images $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. Specifically, it returned the following object information:

1. **Object Categories**, which were sets of labels that described the categories;
2. **Object Region Coordinates**, which were sets of vectors that defined the bounding box coordinates of the objects;
3. **Object Features**, which were sets of vectors that encoded the visual information and were extracted from each object region using RoIAlign [29] (we used ground truth bounding boxes for training and predicted bounding boxes for testing to extract features);
4. **Object Masks (Optional)**, which were sets of $m \times m$ binary masks of the objects that were provided when we used an instance segmentation network.

We used Mask R-CNN [29], which is a widely used instance segmentation network, as the object detection module. Mask R-CNN first extracted feature maps using a CNN and then detected object regions via the following two-stage pipeline. Firstly, the region proposal network (RPN) estimated the coordinates of the bounding boxes and the objectness of the candidate object regions. Next, the second stage estimated the class, bounding box offset and binary mask of each candidate region. During training, the following loss function was used. In the first stage, the smooth-l1 loss of the bounding box coordinates and the binary cross-entropy loss of the objectness were calculated. In the second stage, the smooth-l1 loss of the bounding box coordinates, the cross-entropy loss of the class estimation and the per-pixel binary cross-entropy loss of the binary masks were computed. Finally, the network was trained by performing backpropagation using these calculations. Although we used Mask R-CNN in this study, the proposed network could also use other object detectors, such as those in [25,39–42].

Note that our object detection module was a Siamese network that shared weights between images, which was similar to several related methods [18,21].

3.3. Graph Matching Module

In this section, we describe our graph matching module. The graph matching module used the object features from the object detection module as inputs. Then, it output an assignment matrix $\boldsymbol{P} \in \{0,1\}^{M \times N}$, which represented the object correspondence between the M and N objects in the respective images. The module was based on SuperGlue [34] and had four steps: feature refinement (Section 3.3.1), score matrix prediction (Section 3.3.2), matching prediction (Section 3.3.3) and post-processing (Section 3.3.4).

3.3.1. Feature Refinement

To consider the global scene structure, we constructed a connected graph with objects as its nodes and refined the object features using a graph neural network (GNN). This graph had two types of edges: intra- and inter-image edges (Figure 1). The intra- and inter-image edges were fully connected edges that linked objects within an image and between images, respectively. The GNN updated the node features by alternating between the two types of edges, as follows:

$$ {}^{(l+1)}F_i^1 = {}^{(l)}F_i^1 + f([F_i; m_{\mathcal{E}\to i}]), \tag{2} $$

$$ m_{\mathcal{E}\to i} = \sum_{j:(i,j)\in\mathbb{E}} \mathrm{Softmax}(q_i^\top k_j) v_j, \tag{3} $$

$$ q_i = W_q F_i^1 + b_q, \tag{4} $$

$$ k_i = W_k F_i^1 + b_k, \tag{5} $$

$$ v_i = W_v F_i^1 + b_v, \tag{6} $$

where ${}^{(l)}F_i^1$ is the l-th layer node feature in $\boldsymbol{I}_1$, $\mathcal{E}$ is the set of edges, $[\cdot;\cdot]$ is the vector concatenation and $W_{[q,k,v]}, b_{[q,k,v]}$ are the weights and biases, respectively. The computation of this formula was based on the self-(cross-)attention that is used in transformers [43]. First, the weights were calculated by computing q_i and k_j from the features of the target node i and its neighboring node j, respectively, and applying a Softmax function to the inner product of these vectors. The weights were then used to compute the weighted sum of v_j, which was computed from the neighboring nodes to obtain the message $m_{\mathcal{E}\to i}$. By applying the L-layer GNN to ${}^{(0)}F_i$, we obtained ${}^{(L)}F_i$. The object features in $\boldsymbol{I}_2$ were updated in the same manner. In contrast to [34], we did not use a positional encoder because it did not contribute to performance.

3.3.2. Score Matrix Prediction

Using the node features that were obtained by the GNN, we computed the score matrix $S \in \mathbb{R}^{(M+1)\times(N+1)}$ to compute the assignment matrix that represented the object correspondence between $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. First, we computed $S' = [s_{ij}]_{M\times N,(1<i<M,1<j<N)}$ from the object features, where s_{ij} is the inner product of F_i^1, F_j^2, S_{ij} represents the similarity between the features of each object and S' represents the soft assignment between the objects (Figure 2). Secondly, to account for objects with no matching pairs (i.e., changed objects), we augmented the score matrix to include a dustbin node [34]. The similarity with the dustbin node was set using the trainable parameter z.

Figure 2. A visualization of the affinity scores between the objects in $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. The lines connecting the centers of the boxes represent the s_{ij} (as described in Section 3.3). The figure demonstrates that objects with similar appearances had high affinity scores. For example, in the second image in the first row, the object that is enclosed by the box in $\boldsymbol{I}_1$ and the building in the center of $\boldsymbol{I}_2$ had higher scores.

3.3.3. Matching Prediction

Based on the similarity score, a soft assignment matrix $\bar{\boldsymbol{P}} \in \mathbb{R}^{(M+1)\times(N+1)}$ was predicted using an optimal matching layer, which was based on the Sinkhorn algorithm [37,38]. In the algorithm, an iterative row-wise and column-wise normalization of $\exp(\bar{\boldsymbol{S}})$ was conducted to predict the soft assignments. Notably, this process was differentiable and hence, end-to-end trainable.

3.3.4. Post-Processing

At inference time, the hard assignment $\hat{\boldsymbol{P}}$ was determined using the following process:

$$\hat{\boldsymbol{P}}_{i,j} = \begin{cases} 1 & (\underset{k}{\operatorname{argmax}}\, \bar{\boldsymbol{P}}_{i,k} = j \quad \text{and} \quad \underset{k}{\operatorname{argmax}}\, \bar{\boldsymbol{P}}_{k,j} = i) \\ 0 & (\text{otherwise}) \end{cases} . \tag{7}$$

In contrast to the post-processing in [34], our method did not depend on a specific threshold. Finally, the hard assignments $\boldsymbol{P} \in \{0,1\}^{M\times N}$ were acquired by dropping the dustbin scores from $\hat{\boldsymbol{P}}$ (Figure 3).

Figure 3. A visualization of the object pairs that were obtained by the graph matching module. The red boxes highlight the object in $\boldsymbol{I}_1$ and the corresponding object in $\boldsymbol{I}_2$. When there was no corresponding object, no red boxes are displayed.

3.4. Loss Function

The loss function consisted of two parts: object detection loss and graph matching loss. In the object detection loss, the cross-entropy loss and smooth-l1 loss were used for classification and localization, respectively (see Equation (10) in [25]). Additionally, the per-pixel cross-entropy loss was used when the object masks were estimated. In the graph matching loss, the negative log-likelihood of the ground truth assignment matrix was used [34] (see Equation (20) in [34]).

4. Dataset

Most existing change detection datasets only provide pixel-level change labels (penultimate column in Table 1). To train and evaluate the object-level change detection models, we created a synthetic object-level change detection (SOCD) dataset. This section describes the creation process of the SOCD dataset and the dataset features.

Table 1. A comparison of the existing street view change detection datasets. The *Viewpoint Difference* column indicates the scale of the viewpoint differences between image pairs. Notably, the SOCD dataset included four separate scales for viewpoint differences, which enabled an examination of the relationship between the scale of viewpoint differences and the accuracy of the change detection model. The *Annotation* columns indicate each the annotation type that was used in each dataset (i.e., pixel-level annotation, such as binary change masks or semantic change masks, and object-level annotation, such as bounding boxes or instance masks).

Dataset	Domain	Image Type	Viewpoint Difference	Number of Pairs	Number of Semantic Classes	Annotation	
						Pixel-Level	Object-Level
PCD (TSUNAMI) [27]	Real	Panoramic	Small	100	- *	✓	
PCD (GSV) [27]	Real	Panoramic	Small	100	- *	✓	
VL-CMU-CD [13]	Real	Perspective	Small †	1362	- *	✓	
CDnet2014 [44]	Real	Perspective	None	70,000	- *	✓	
SOCD (Ours)	Synthetic	Perspective	None∼Large (Four different viewpoint scales)	15,000	4	✓	✓

* No semantic labels; † image pairs in the VL-CMU-CD dataset were densely aligned using depths that were reconstructed using stereo matching.

The SOCD dataset was created using a CARLA simulator in order to research and develop self-driving vehicles. The simulator provided virtual maps that simulated urban areas. Objects, such as cars and buildings, could be placed freely within the environment. In addition, users could virtually place cameras at arbitrary locations within the environment and capture realistic images that corresponded to the camera positions using the rendering engine of the simulator. The simulator was equipped with various sensors and could acquire not only RGB images but also depth images, semantic (pixel-wise) labels and instance masks.

There were three advantages to using a simulator to create our dataset. Firstly, it allowed the dataset to be created rapidly. When creating a real image dataset, it is necessary to capture images over long periods of time, such as several months or years, in order to observe changes. Manual labeling is also very time-consuming, so the use of a simulator reduced the time that was required because the ground truth images were obtained automatically. Secondly, the cost of creating images and labels is low. Because there was no need to drive vehicles around to capture the images or perform manual labeling, we could create a large dataset at a low cost (the fifth column of Table 1). Finally, it was possible to control the camera position within the environment. Leveraging this advantage, we investigated the effects of the magnitude of viewpoint differences between images on change detection performance.

Figure 4 illustrates the dataset creation process. As shown in Figure 4, the cameras were first placed at random street locations, where the images at t_1 were rendered. Next, objects were added and deleted to create environmental changes. Then, by placing the

camera at the same position as that in the t_1, the images at t_2 were captured. This process was repeated in multiple environments, which resulted in 15,000 image pairs. The field of view of the cameras was set to 90°. Four object categories were selected: buildings, cars, poles and traffic signs/lights. For each object, the following annotations were obtained, as shown in Figure 5: bounding boxes, instance masks, correspondence between objects, pixel-level changes and object-level changes. In contrast to the traditional change detection datasets, object-level changes were available.

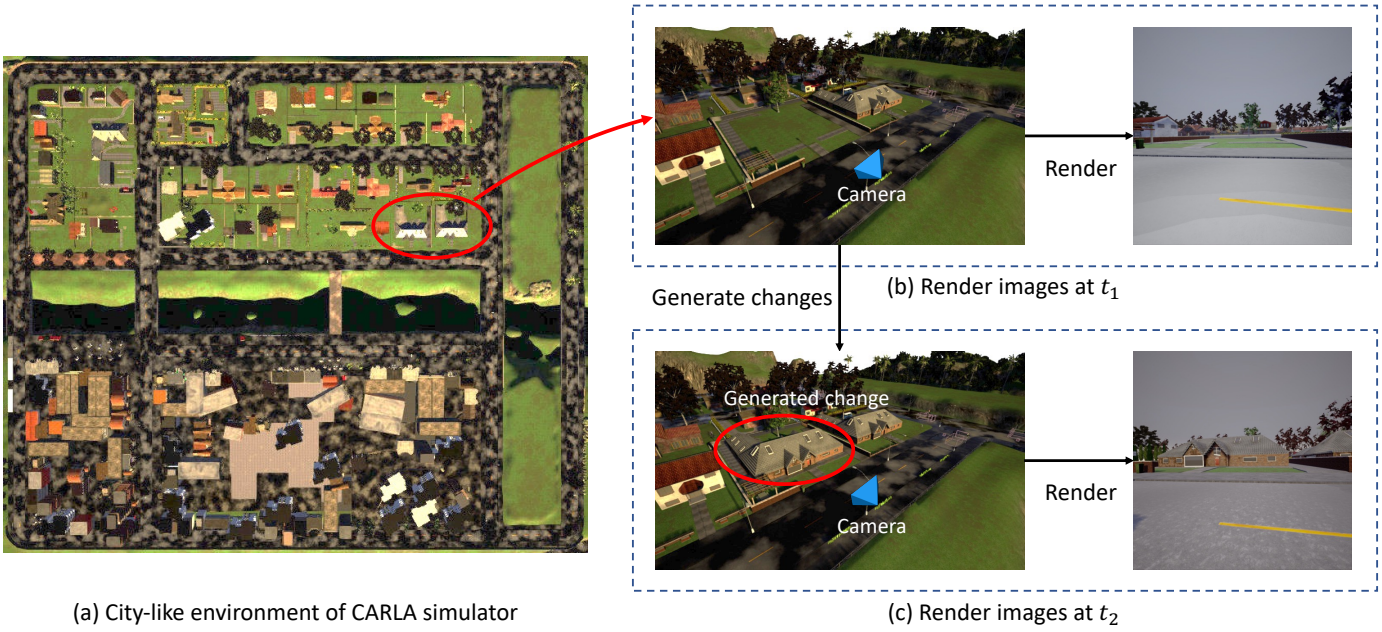

Figure 4. An overview of the data acquisition pipeline. The images were rendered from cameras that were set within the CARLA simulator environment.

To investigate the robustness of the change detection model to viewpoint differences, we rendered the image pairs with varying viewpoint differences (the fourth column of Table 1). Specifically, the dataset had four categories $\{S_1, S_2, S_3, S_4\}$ with varying yaw angle differences. In S_1, there was no difference in yaw angle. In $S_{2,3,4}$, the difference in yaw angle was uniformly sampled within the ranges of $[0, 10]$, $[10, 20]$ and $[20, 30]$, respectively. The rolls and pitches were randomly shifted within the range of $[0°, 5°]$ for all categories and the horizontal translation was randomly shifted within the range of $[0, 1.0]$ m. Figure 6 shows a histogram of the intersection over union (IoU) values of the paired bounding boxes. When the viewpoint differences increased, the IoU values gradually decreased, which indicated mild to severe misalignment between the objects. Figure 7 shows some examples of image pairs from the dataset that had viewpoint differences. With such a wide degree of misalignment, the robustness of the change detection models could be thoroughly evaluated in a controlled manner.

Figure 5. Examples of the instance masks and bounding boxes for the objects/changes. The light blue is buildings, pink is cars, green is poles.

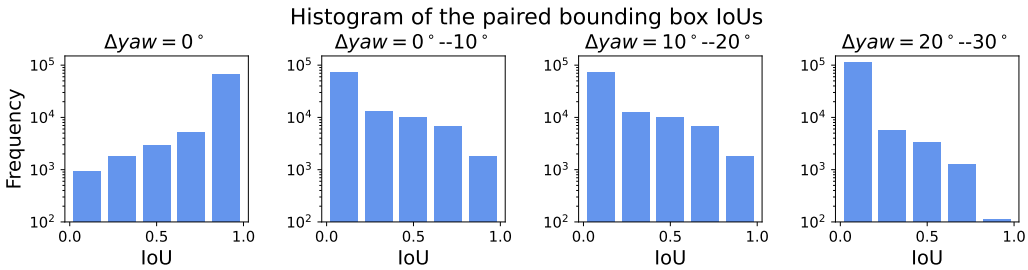

Figure 6. The histograms of the IoU values of the paired bounding boxes in the SOCD dataset and the OCD dataset. In the SOCD dataset, the larger the viewpoint differences between an image pair, the smaller the overlap of objects. By contrast, the OCD dataset had a wide distribution of IoU values because of the various viewpoint differences between the image pairs.

Figure 7. Examples from the SOCD dataset. I_1 are images before any changes and I_2^i are the images after the changes. For $i = 1$, there were no viewpoint differences between I_1 and I_2^i. For $i = 2, 3, 4$, a viewpoint difference existed between I_1 and I_2^i. The scale of the difference in yaw angle was different for $i = 2, 3$ and 4 and ranged from 0° to 10°, from 10° to 20° and from 20° to 30°, respectively.

5. Experiments

To demonstrate the feasibility and effectiveness of our object-level change detection network, we conducted experiments using the SOCD dataset. Section 5.1 explains the experimental setup and Section 5.2 describes a comparison between the proposed network and existing state-of-the-art methods.

5.1. Experimental Settings

5.1.1. Dataset Settings

First, we divided the simulator map into four sections (vertically and horizontally) and designated two non-adjacent sections as the training/validation and testing sets. To avoid overlapping the fields of view of the training/validation set and the testing set, only cameras that did not face the boundary lines were selected in the vicinity of the division lines. Finally, the dataset was divided into training, validation and testing sets in a ratio of 40:10:50. The validation data were used to determine the hyperparameters, such as the number of GNN layers and the learning rate.

5.1.2. Implementation Details

We used Faster R-CNN [25] and Mask R-CNN [29] in a feature pyramid network [45] with a ResNet-101 [46] backbone. The graph matching module had 15 GNN layers. We trained our proposed network in two stages: pre-training the object detection module and fine-tuning the entire network. First, during pre-training using a labeled dataset, the Mask R-CNN was trained for 26 epochs. The batch size was eight, the image size was 800 $\times$ 800 pixels and the images were flipped horizontally with a probability of 0.5. The learning rate was initially set as 0.002 and was multiplied by 0.1 for both the 16th and 22nd epochs. Next, we trained the entire network for 20 epochs during the fine-tuning step. The batch size was set to four and the image size was 800 $\times$ 800 pixels. The learning rate was initially set as 0.001 and was multiplied by 0.1 at the 10th epoch. We used four NVIDIA Tesla V100 GPUs during both steps and implemented the proposed network in PyTorch. The learning strategy was distributed in parallel.

5.1.3. Comparison to Other Methods

To confirm the effectiveness of the proposed method, we compared it to the following three methods:

1. *Pixel-Level Change Detection Methods*: Most of the existing methods could not be directly compared to our method because they detected pixel-level changes rather than object-level changes. To compare our method to existing pixel-level change detection networks, we augmented the existing methods by adding post-processing that converted pixel-level change masks into instance masks or bounding boxes. Specifically, the change masks in the existing methods were clipped using the outputs of the pre-trained Mask R-CNN. This clipping was performed when more than 50% of the instance mask overlapped the change mask in the experiment. We used ChangeNet [47] and CSSCDNet [18] as the pixel-level networks;
2. *Our Method Without the Graph Matching Module*: A model without the graph matching module was tested using the proposed method. This model created a matrix whose elements were the distances between object features in $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$ and then found matches in post-processing, which was equivalent to that described in Section 3.3.4;
3. *Our Method With a GT Mask*: A model that used the correct bounding boxes instead of the bounding boxes that were extracted by the object detector was also tested. We used the same parameters for the feature extractor and the graph matching module as those that were used for the proposed trained network.

5.1.4. Evaluation Metrics

Although *pixel-level* metrics, such as mean pixel accuracy or mean Intersection over Union (for more detail, see Section 5 in [48]), have been widely used in previous studies [49], these metrics were not suitable to evaluate *object-level* scene change detection. For pixel-level metrics, errors in small objects have little impact and they are often neglected compared to larger objects, which was not desirable for testing object-level scene change detection. Therefore, to evaluate the performance of the change detection methods in an object-centric manner, we employed the F_1 score, which is widely used in object detection tasks.

The F_1 score was calculated as follows. The ground truth (GT) regions with the largest IoU values were assigned for each predicted region. When the IoU value was greater than 0.5, the predicted region was classified as a true positive (TP) or false positive (FP). Any GT regions that the model failed to detect were classified as false negatives (FN). We reported the performance of object-level change detection as the following F_1 score:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{8}$$

where:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

5.2. Experimental Results

In Table 2, we present the results of the object-level change detection methods using the SOCD dataset. When there were no viewpoint differences between the images, the baseline method with CSSCDNet produced the best score. In the case of crowdsourcing and mobile cameras, this situation would be uncommon. By contrast, when there were viewpoint differences between the images, the proposed method outperformed all of the baseline methods and improved the score by more than 40%. This result indicated the robustness of the proposed approach to viewpoint differences. In addition, we evaluated the proposed method using correct instance masks during inference. The results that are shown in the last row of the table suggested that the performance of the proposed method could be further improved by employing a more powerful object detector, such as DyHead [42], which is a state-of-the-art object detection network.

Table 2. Our evaluation of object-level change detection methods using the SOCD dataset. Our proposed network outperformed all baseline methods in terms of F_1 score when viewpoint differences existed.

Method	**Viewpoint Difference**			
	$\Delta yaw = 0°$	$\Delta yaw = 0$–$10°$	$\Delta yaw = 10$–$20°$	$\Delta yaw = 20$–$30°$
Pixel-level Methods + Post-Processing				
ChangeNet [47]	0.219	0.142	0.155	0.133
CSSCDNet [18] (Without correction layer)	0.453	0.234	0.241	0.205
CSSCDNet [18] (With correction layer)	**0.508**	0.274	0.258	0.208
Object-level Methods				
Ours (Without graph matching module)	0.406	0.337	0.335	0.295
Ours	0.463	**0.401**	**0.396**	**0.354**
Ours (With GT mask)	0.852	0.650	0.652	0.546

Figure 8 shows some of the results from our qualitative evaluation. Our network successfully detected object-level changes, even when viewpoint differences existed between the images (see from the second to sixth rows). The proposed method also successfully detected thin objects, such as street light poles (first row), and small objects, such as traffic signs (fifth row), which the existing methods failed to detect. Furthermore, it could be observed that the change masks captured the shapes of objects more accurately than those in the existing methods, particularly when the scale of the viewpoint difference was relatively large (fourth, fifth and sixth rows).

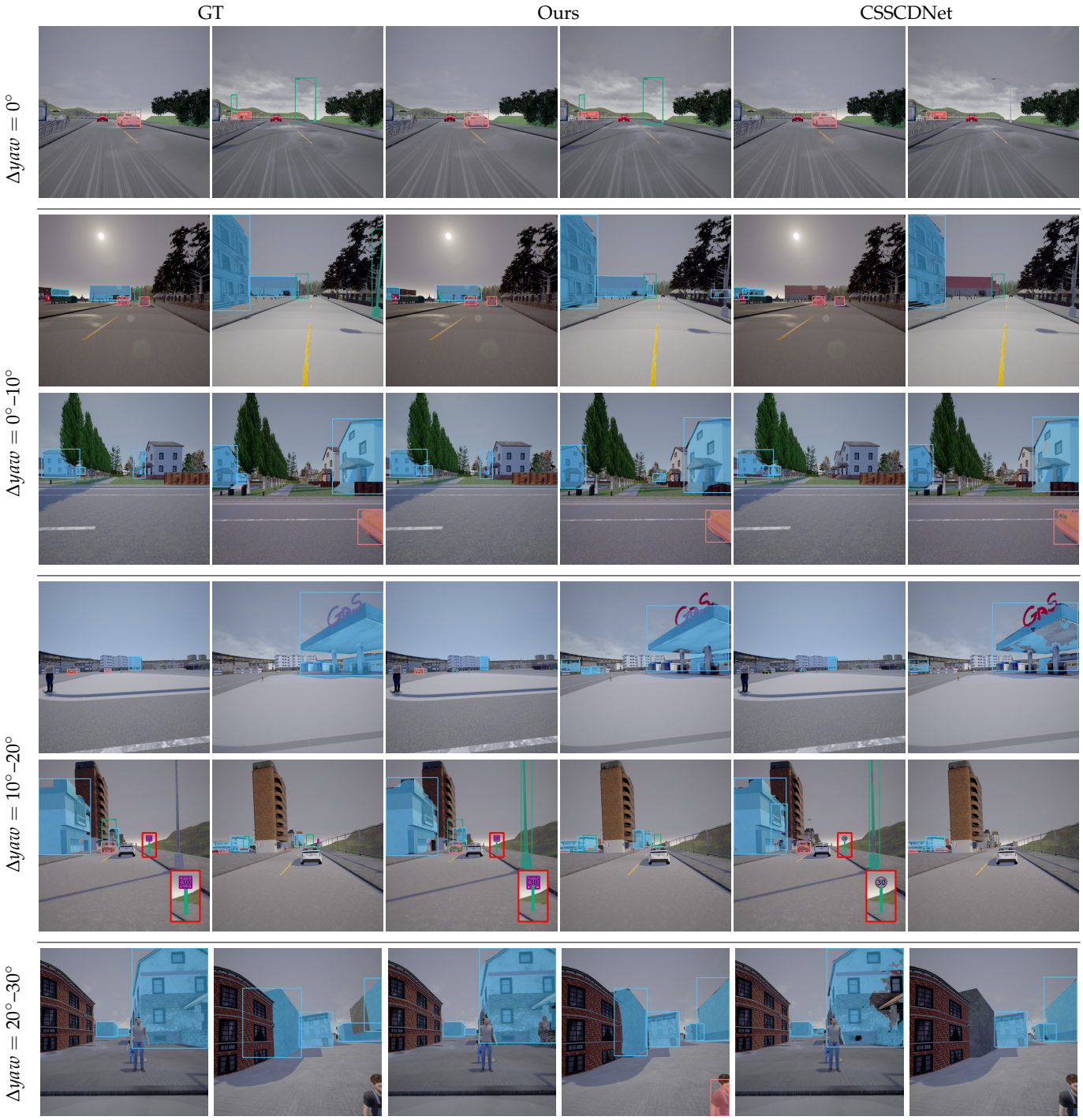

Figure 8. The qualitative results from using the SOCD dataset. Object-level changes indicated by color-coded bounding boxes: light blue for buildings, pink for cars, green for poles and purple for traffic signs. Our network was good at locating object-level changes in image pairs.

6. Discussion

In this section, we present a more detailed discussion about the properties and challenges of the proposed method. Firstly, we discuss the differences between the proposed network and the existing methods in Section 6.1. Then, we provide a quantitative and qualitative analysis of the errors in Section 6.2. Finally, we describe the future directions of our work in Section 6.3.

6.1. Comparison to Existing Methods

Figure 9 shows the precision, recall and F_1 scores of the proposed method and the existing methods for each scale of viewpoint difference. Firstly, the values of precision decreased as the yaw angle between the images increased for all methods. However, the recall scores for the existing methods decreased as the yaw angle between the images increased, whereas the score for the proposed method remained unchanged. This difference in trend was reflected in the final F_1 score and, therefore, the proposed method outperformed the conventional methods when there were viewpoint differences. In other words, the proposed method had an advantage over the existing methods because it did not "miss" many changes, even when viewpoint differences existed. This property would be important for various applications, such as infrastructure inspection, because missing changes (e.g., aging or damage to infrastructure) could lead to hazards.

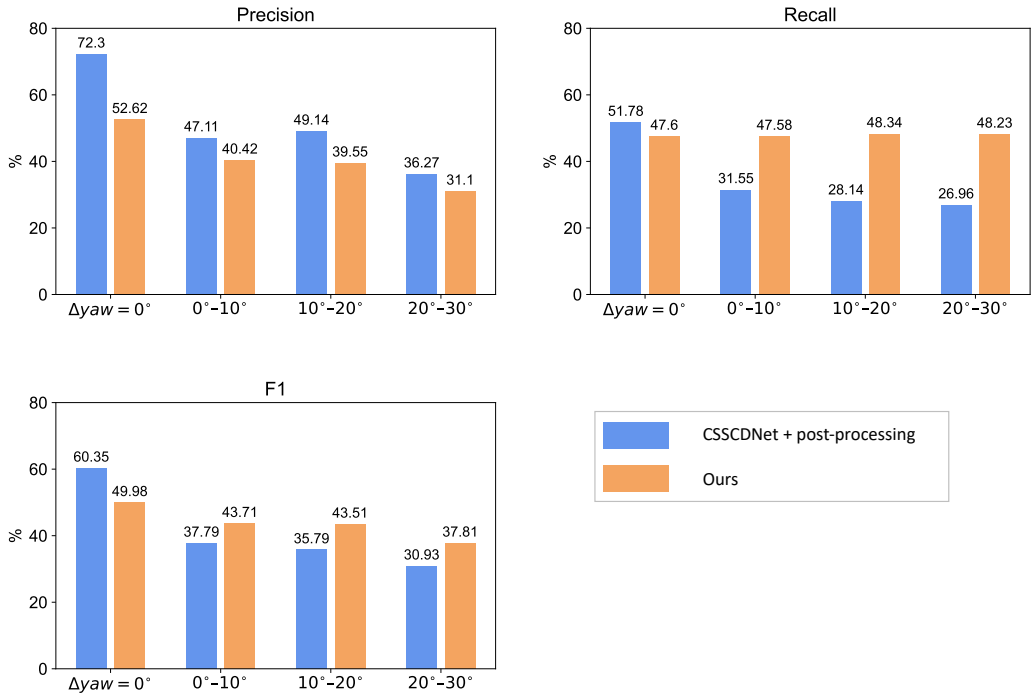

Figure 9. The precision, recall and F_1 scores of our proposed network and the existing methods [18] for the separate scales of viewpoint differences between the images.

6.2. Error Analysis

The proposed method still presented scope for improvement and a non-negligible number of errors occurred. In this subsection, we present a qualitative and quantitative analysis of these errors in the proposed network. First, we discuss three typical failures that occurred with our proposed method. The first major failure was caused by an error in the object detection module. As shown in the first row in Figure 10, the proposed method failed to detect a building in the center of the screen in $\boldsymbol{I}_2$ and detected a single building with multiple instance masks. Thus, this error in object detection directly led to an error in change detection. The second failure was a matching error. As shown in the second row in Figure 10, even though the proposed method could detect the same vehicle in $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, it failed to match it and incorrectly detected it as a change. The third failure was an error that was caused by recognizing objects that moved out of the field of view as changes. A traffic sign was on the left edge of $\boldsymbol{I}_1$ (third row in Figure 10); however, in $\boldsymbol{I}_2$, the traffic sign that should have remained unchanged was obscured. The proposed method incorrectly detected the transition of objects out of the field of view as a change.

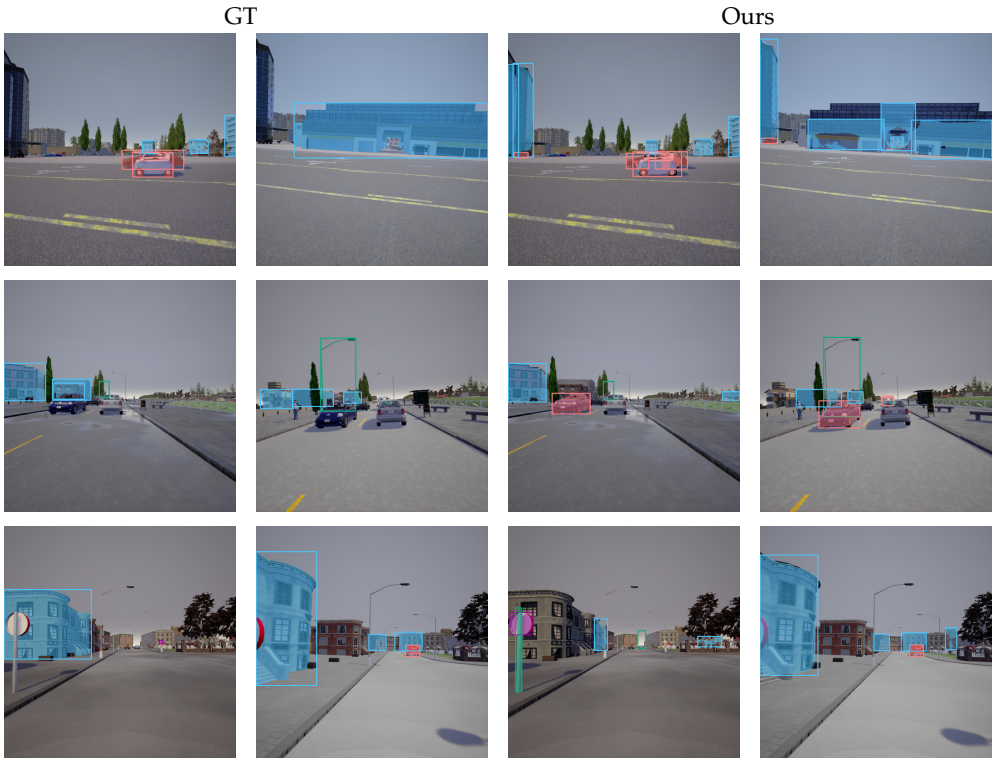

Figure 10. Examples of failures. Object-level changes indicated by color-coded bounding boxes: light blue for buildings, pink for cars, green for poles and purple for traffic signs. The examples in the first and second rows suffered from errors in the object detection module and the graph matching module, respectively. The last row shows the error that was caused by moving an object out of the field of view.

Next, we present a quantitative analysis of the errors in the proposed network. Figure 11 presents a summary of the errors in the proposed method using the testing data for each viewpoint difference scale. Errors were classified into the following three categories: (1) errors that were due to recognizing objects that moved out of the field of view as changes (FP (out of view)); (2) false positives (FP) other than FP (out of view); and (3) false negatives. FP (out of view) included the errors that are shown in Figure 8 and in the third row of Figure 10. Figure 11 shows that when there were no viewpoint differences between the images (i.e., the bar for $\Delta yaw = 0°$), FP and FN results accounted for all of the errors. In this case, the number of errors could be reduced by improving the performance of the object detection and graph matching modules. However, when there were viewpoint differences between the images (i.e., the bars for 0°–10°, 10°–20° and 20°–30° in Figure 11), the only errors that occurred were classified as FP (out of view). Furthermore, as the viewpoint difference increased, FP (out of view) errors accounted for a more significant percentage of the total errors. Section 6.3 describes some possible countermeasures against objects moving out of the field of view.

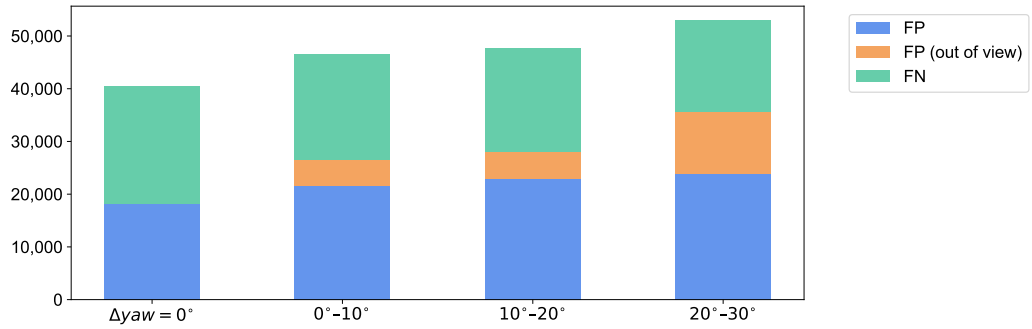

Figure 11. A summary of the errors in the proposed network. The bars show the percentage of each error in the proposed method for the separate scales of viewpoint differences between the images. *FP* denotes false positive results (except for *FP (out of view)* as *FP (out of view)* represents false positive results that occurred because of an object moving out of the field of view) and *FN* represents false negative results.

6.3. Limitations and Future Directions

In this section, we describe several limitations of this method and directions for future research. Firstly, continuous objects that bounding boxes could not surround, such as roads or white lines, were beyond the scope of this work. Therefore, it is a challenge for future work to simultaneously handle object-level changes in countable objects and pixel-level changes in continuous objects. This task corresponds to panoptic segmentation [50–54] and we believe that *panoptic change detection* research is a promising direction for the remote sensing community.

Secondly, our proposed network suffered from errors that were caused by the movement of objects to/from the field of view (last row in Figure 8). In principle, this error would be a difficult problem to address using current matching-based change detection. One possible countermeasure could be to use images with broader fields of view; however, this would limit the applicability of the proposed network. Therefore, to solve this problem, we will investigate the possible use of depth information. By using the depth information of objects, we could estimate where an object in one image is located in the paired image. Hence, we plan to extend our network to include monocular 3D bounding box estimation, which could improve the overall performance of our object-level change detection method.

Thirdly, the proposed method could not handle changes other than the appearance or disappearance of objects. For example, various types of changes can occur to objects in images, such as deformation, discoloration and changes in position or posture. However, the current framework could not recognize these various categories of change. Therefore, in the future, it is necessary to expand the dataset and method to include various changes. In addition, by assuming that all types of change could occur, we plan to express the content of changes in natural language using image captioning technology [55–60].

Finally, there was a lack of verification using real data. There are several difficulties in creating a dataset that includes all of the possible changes in real urban environments. Firstly, many years of observation in urban areas would be required to monitor changes in the real environments. In addition, to observe enough changes to train a model, a large area must be observed and a large amount of image data must be acquired. Moreover, the creation of object-level annotations is time-consuming and labor-intensive. Therefore, in this study, we only performed validation using synthetic data. Now that we know that we can use synthetic data to some extent, we plan to construct a real image dataset for future further validation.

7. Conclusions

We proposed a new change detection network to detect object-level changes in image pairs with viewpoint differences. The proposed framework effectively combined an object detection module and a graph matching module to detect object-level changes in image

pairs accurately and robustly, as shown in Figure 1. The object detection module extracted object regions in the form of bounding boxes or instance masks. The graph matching module considered the objects in each image as a graph and built correspondences between the objects in the image pairs. To verify the effectiveness of the proposed network, we created an SOCD dataset using the CARLA simulator, which was the first dataset with object-level change labels. Our experimental results using the SOCD dataset demonstrated that our proposed framework was superior to existing methods, as shown in Table 2 and Figure 8. We also analyzed our experimental results to gain a deeper understanding of the properties of the proposed network. The analysis revealed that the proposed method did not suffer from loss of recall, even when viewpoint differences increased (Figure 9) and several typical failures were identified (Figure 10). The main failures could be categorized as object detection errors, graph matching errors and errors that were caused by objects moving out of the field of view. Future directions of study include improving the performance of each module by devising new network structures and learning methods and introducing 3D object detection to handle objects that move out of the field of view.

Author Contributions: Conceptualization, K.D. and K.S.; methodology, K.D. and K.S.; validation, K.D. and R.H.; writing—original draft preparation, K.D.; writing—-review and editing, K.D., R.H., Y.I., M.O., Y.M. and K.S.; supervision, Y.I., Y.M., M.O. and K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by JSPS KAKENHI (grant number: 20H04217).

Data Availability Statement: The SOCD dataset will be publicly available at https://doiken23.github.io/object_level_cd accessed on 17 August 2022.

Acknowledgments: The authors are grateful to Rio Yokota and Shun Iwase for useful discussions and for providing the image data and dataset creation codes. We also thank Yukuko Furukawa for valuable discussions at usual meetings.

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

References

1. Lu, D.; Mausel, P.; Brondízio, E.; Moran, E. Change detection techniques. *Int. J. Remote Sens.* **2004**, *25*, 2365–2401. [CrossRef]
2. Radke, R.J.; Andra, S.; Al-Kofahi, O.; Roysam, B. Image Change Detection Algorithms: A Systematic Survey. *IEEE Trans. Image Process.* **2005**, *14*, 294–307. [CrossRef] [PubMed]
3. Bai, B.; Fu, W.; Lu, T.; Li, S. Edge-Guided Recurrent Convolutional Neural Network for Multitemporal Remote Sensing Image Building Change Detection. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5610613. [CrossRef]
4. Wahl, D.E.; Yocky, D.A.; Jakowatz, C.V.; Simonson, K.M. A New Maximum-Likelihood Change Estimator for Two-Pass SAR Coherent Change Detection. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 2460–2469. [CrossRef]
5. Wu, C.; Du, B.; Zhang, L. Slow Feature Analysis for Change Detection in Multispectral Imagery. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 2858–2874. [CrossRef]
6. Liu, S.; Bruzzone, L.; Bovolo, F.; Zanetti, M.; Du, P. Sequential Spectral Change Vector Analysis for Iteratively Discovering and Detecting Multiple Changes in Hyperspectral Images. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 4363–4378. [CrossRef]
7. Hazel, G. Object-level change detection in spectral imagery. *IEEE Trans. Geosci. Remote Sens.* **2001**, *39*, 553–561. [CrossRef]
8. Celik, T. Change Detection in Satellite Images Using a Genetic Algorithm Approach. *IEEE Geosci. Remote Sens. Lett.* **2010**, *7*, 386–390. [CrossRef]
9. Benedek, C.; Sziranyi, T. Change Detection in Optical Aerial Images by a Multilayer Conditional Mixed Markov Model. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 3416–3430. [CrossRef]
10. Chen, B.; Chen, Z.; Deng, L.; Duan, Y.; Zhou, J. Building change detection with RGB-D map generated from UAV images. *Neurocomputing* **2016**, *208*, 350–364. [CrossRef]
11. Feurer, D.; Vinatier, F. Joining multi-epoch archival aerial images in a single SfM block allows 3-D change detection with almost exclusively image information. *ISPRS J. Photogramm. Remote Sens.* **2018**, *146*, 495–506. [CrossRef]
12. Taneja, A.; Ballan, L.; Pollefeys, M. Image Based Detection of Geometric Changes in Urban Environments. In Proceedings of the IEEE International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2336–2343. [CrossRef]
13. Alcantarilla, P.F.; Stent, S.; Ros, G.; Arroyo, R.; Gherardi, R. Street-View Change Detection with Deconvolutional Networks. *Auton. Robot.* **2018**, *42*, 1301–1322. [CrossRef]

14. Palazzolo, E.; Stachniss, C. Fast Image-Based Geometric Change Detection Given a 3D Model. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–26 May 2018; pp. 6308–6315. [CrossRef]
15. Jo, K.; Kim, C.; Sunwoo, M. Simultaneous Localization and Map Change Update for the High Definition Map-based Autonomous Driving Car. *Sensors* **2018**, *18*, 3145. [CrossRef] [PubMed]
16. Pannen, D.; Liebner, M.; Burgard, W. HD map change detection with a boosted particle filter. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 2561–2567. [CrossRef]
17. Furukawa, Y.; Suzuki, K.; Hamaguchi, R.; Onishi, M.; Sakurada, K. Self-supervised Simultaneous Alignment and Change Detection. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 6025–6031.
18. Sakurada, K.; Shibuya, M.; Wang, W. Weakly Supervised Silhouette-based Semantic Scene Change Detection. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Online, 31 May–31 August 2020; pp. 6861–6867.
19. Pannen, D.; Liebner, M.; Hempel, W.; Burgard, W. How to Keep HD Maps for Automated Driving Up To Date. In Proceedings of the International Conference on Robotics and Automation (ICRA), online, 31 May–31 August 2020; pp. 2288–2294.
20. Heo, M.; Kim, J.; Kim, S. HD Map Change Detection with Cross-Domain Deep Metric Learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 10218–10224.
21. Lei, Y.; Peng, D.; Zhang, P.; Ke, Q.; Li, H. Hierarchical Paired Channel Fusion Network for Street Scene Change Detection. *IEEE Trans. Image Process.* **2020**, *30*, 55–67. [CrossRef]
22. Wang, Q.; Zhang, X.; Chen, G.; Dai, F.; Gong, Y.; Zhu, K. Change detection based on Faster R-CNN for high-resolution remote sensing images. *Remote Sens. Lett.* **2018**, *9*, 923–932. [CrossRef]
23. Ji, S.; Shen, Y.; Lu, M.; Zhang, Y. Building Instance Change Detection from Large-Scale Aerial Images using Convolutional Neural Networks and Simulated Samples. *Remote Sens.* **2019**, *11*, 1343. [CrossRef]
24. Zhang, L.; Hu, X.; Zhang, M.; Shu, Z.; Zhou, H. Object-level change detection with a dual correlation attention-guided detector. *ISPRS J. Photogramm. Remote Sens.* **2021**, *177*, 147–160. [CrossRef]
25. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In Proceedings of the Advances in Neural Information Processing Systems 28, Montreal, QC, Canada, 7–10 September 2015; pp. 91–99.
26. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning, Mountain View, CA, USA, 13–15 November 2017; Volume 78, pp. 1–16.
27. Sakurada, K.; Okatani, T. Change Detection from a Street Image Pair using CNN Features and Superpixel Segmentation. In Proceedings of the British Machine Vision Conference (BMVC), Swansea, UK, 7–10 September 2015; pp. 61.1–61.12. [CrossRef]
28. Guo, E.; Fu, X.; Zhu, J.; Deng, M.; Liu, Y.; Zhu, Q.; Li, H. Learning to Measure Changes: Fully Convolutional Siamese Metric Networks for Scene Change Detection. *arXiv* **2018**, arXiv:1810.09111.
29. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
30. Conte, D.; Sansone, C.; Vento, M.; Conte, D.; Foggia, P.; Sansone, C.; Vento, M. Thirty Years Of Graph Matching In Pattern Recognition. *Int. J. Pattern Recognit. Artif. Intell.* **2004**, *18*, 265–298. [CrossRef]
31. Yan, J.; Yang, S.; Hancock, E. Learning for Graph Matching and Related Combinatorial Optimization Problems. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, 7–15 January 2020; pp. 4988–4996. [CrossRef]
32. Zanfir, A.; Sminchisescu, C. Deep Learning for Graph Matching. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–22 June 2018; pp. 2684 – 2693.
33. Wang, R.; Yan, J.; Yang, X. Learning Combinatorial Embedding Networks for Deep Graph Matching. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
34. Sarlin, P.E.; Detone, D.; Malisiewicz, T.; Rabinovich, A.; Zurich, E. SuperGlue: Learning Feature Matching with Graph Neural Networks. In Proceedings of the EEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online 14–19 June 2020; pp. 4938–4947.
35. Battaglia, P.W.; Hamrick, J.B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; et al. Relational inductive biases, deep learning, and graph networks. *arXiv* **2018**, arXiv:1806.01261.
36. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural Message Passing for Quantum Chemistry. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, NSW, Australia, 6–11 August 2017; pp. 1263–1272.
37. Cuturi, M. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; Volume 26, pp. 2292–2300.
38. Sinkhorn, R.; Knopp, P. Concerning Nonnegative Matrices and Doubly Stochastic Matrices. *Pac. J. Math.* **1967**, *21*, 343–348. [CrossRef]
39. Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.

40. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In Proceedings of the 16th European Conference, Glasgow, UK, 23–28 August 2020; pp. 213–229.
41. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021; pp. 13029–13038.
42. Dai, X.; Chen, Y.; Xiao, B.; Chen, D.; Liu, M.; Yuan, L.; Zhang, L. Dynamic Head: Unifying Object Detection Heads with Attentions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 19–25 June 2021.
43. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.U.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
44. Wang, Y.; Jodoin, P.M.; Porikli, F.; Konrad, J.; Benezeth, Y.; Ishwar, P. CDnet 2014: An Expanded Change Detection Benchmark Dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 387 – 394.
45. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125. [CrossRef]
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
47. Varghese, A.; Gubbi, J.; Ramaswamy, A. ChangeNet: A Deep Learning Architecture for Visual Change Detection. In Proceedings of the ECCV Workshop, Munich, Germany, 8–14 September 2018.
48. Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3523–3542. [CrossRef] [PubMed]
49. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Garcia-Rodriguez, J. A Review on Deep Learning Techniques Applied to Semantic Segmentation. *arXiv* **2017**, arXiv:1704.06857.
50. Kirillov, A.; He, K.; Girshick, R.; Rother, C.; Dollar, P. Panoptic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
51. Cheng, B.; Collins, M.D.; Zhu, Y.; Liu, T.; Huang, T.S.; Adam, H.; Chen, L.C. Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 13–19 June 2020.
52. Li, Y.; Zhao, H.; Qi, X.; Wang, L.; Li, Z.; Sun, J.; Jia, J. Fully Convolutional Networks for Panoptic Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 20–25 June 2021; pp. 214–223.
53. Wang, H.; Zhu, Y.; Adam, H.; Yuille, A.; Chen, L.C. MaX-DeepLab: End-to-End Panoptic Segmentation With Mask Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Online, 20–25 June 2021; pp. 5463–5474.
54. Li, Z.; Wang, W.; Xie, E.; Yu, Z.; Anandkumar, A.; Alvarez, J.M.; Luo, P.; Lu, T. Panoptic SegFormer: Delving Deeper Into Panoptic Segmentation With Transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 19–24 June 2022; pp. 1280–1289.
55. Vinyals, O.; Toshev, A.; Bengio, S.; Erhan, D. Show and tell: A neural image caption generator. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3156–3164.
56. Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; Bengio, Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015, Volume 37, pp. 2048–2057.
57. Anderson, P.; He, X.; Buehler, C.; Teney, D.; Johnson, M.; Gould, S.; Zhang, L. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
58. Jhamtani, H.; Berg-Kirkpatrick, T. Learning to Describe Differences Between Pairs of Similar Images. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4024–4034. [CrossRef]
59. Park, D.H.; Darrell, T.; Rohrbach, A. Robust Change Captioning. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 4624–4633.
60. Qiu, Y.; Yamamoto, S.; Nakashima, K.; Suzuki, R.; Iwata, K.; Kataoka, H.; Satoh, Y. Describing and Localizing Multiple Changes With Transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Online, 11–17 October 2021; pp. 1971–1980.