This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation. Except for this watermark, it is identical to the accepted version; the final published version of the proceedings is available on IEEE Xplore.

The Change You Want to See (Now in 3D)

Ragav Sachdeva Andrew Zisserman

Visual Geometry Group, Dept. of Engineering Science, University of Oxford

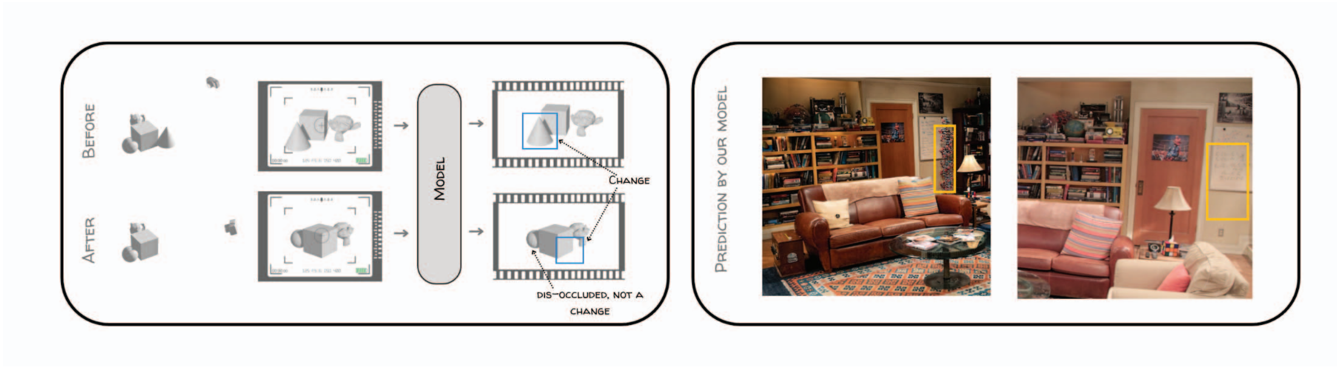


Figure 1. **On the left:** An illustration of the problem we aim to address in this work. Given an image pair of a 3D scene, with a significant shift in camera pose, can we detect what has changed? **On the right:** Top prediction by our model on images from the popular TV show *The Big Bang Theory* depicting the missing DNA-model.

Abstract

The goal of this paper is to detect what has changed, if anything, between two "in the wild" images of the same 3D scene acquired from different camera positions and at different temporal instances. The open-set nature of this problem, occlusions/dis-occlusions due to the shift in viewpoint, and the lack of suitable training datasets, presents substantial challenges in devising a solution.

To address this problem, we contribute a change detection model that is trained entirely on ***synthetic data*** *and is* ***class-agnostic****, yet it is* ***performant out-of-the-box on real world images*** *without requiring fine-tuning. Our solution entails a "register and difference" approach that leverages self-supervised frozen embeddings and feature differences, which allows the model to generalise to a wide variety of scenes and domains. The model is able to* ***operate directly on two RGB images****, without requiring access to ground truth camera intrinsics, extrinsics, depth maps, point clouds, or additional before-after images. Finally, we collect and* ***release a new evaluation dataset*** *consisting of real-world image pairs with human-annotated differences and demonstrate the efficacy of our method. The code, datasets and pre-trained model can be found at:* `https://github.com/ragavsachdeva/CYWS-3D`

1. Introduction

From the way leaves rustle in the wind to the shifting patterns of clouds in the sky, our world is in a constant state of flux. Yet detecting and localising changes in complex 3D scenes remains a challenging task for computer vision. Imagine being able to identify the changes between two images of the same scene captured at separate moments in time and from different viewpoints, as shown in Figure 1. This is the challenge we aim to address in this work. With applications in fields such as robotics, facility monitoring, forensics and augmented reality, our work has the potential to unlock new possibilities for understanding and interacting with our dynamic world.

We formulate the problem we study as follows: Given a pair of 2D images of a 3D scene, captured with a significant shift in camera position and at different temporal instances, we wish to localise the changes between them, if any. In particular, we wish to capture *everything* that is physically different **in the regions that are visible in both the images** while disregarding areas that appear or disappear from view due to the shift in camera pose or occlusion. This includes objects that may have been added or removed from the scene, and text or decorations that may have been added to an object, but we wish to ignore photometric differences such as a lighting change. Under this setting, differentiating true changes from occlusions or dis-occlusions is intrinsically ill-posed using 2D images alone. In other words, answering the question “Is this object missing in the other image, hidden behind another object, or simply out of frame?” fundamentally requires the 3D shape of the scene, which is not directly available to us apriori. Furthermore, it is not possible to compute the shape of the given 3D scene using two-view stereo methods as they rely on corresponding points in the two images which are inherently non-existent in case of changes such as missing objects. Consequently, in the absence of the scene’s ground truth geometry, it is theoretically infeasible to reason about the relative position, scale and shape of the objects in the scene, and how or where they might appear when observed from a different viewpoint (see Figure 1). This, coupled with the lack of large-scale real-world datasets that include image pairs of changing scenes captured from significantly different viewpoints, makes devising a solution to this general two-view change detection problem very challenging.

Nevertheless, our objective is to detect changes in “in the wild” real world images with minimal constraints and operate on RGB images only, without assuming access to ground truth geometry, depth, camera parameters, camera poses etc. Our solution is to build on the insight that once the two views are *registered*, it is relatively easy to determine what has changed. We thus proceed in two stages: (1) *register* by warping the spatial features from one image to the other, and this fundamentally needs to be in 3D; (2) *determine the differences* by training a detector on top of these registered spatial feature maps in order to identify the significant changes, and ignore “nuisance” variables such as changes in lighting. Briefly, we first use an off-the-shelf pre-trained visual backbone to extract spatial feature maps for an RGB image pair. We then lift these 2D feature maps to 3D by making use of state-of-the-art monocular depth estimation models, followed by a differentiable feature registration module (DFRM) to align and render the features maps back to 2D in the other view. Finally, we utilise a simple detection head to process these features and output the changes. To overcome the issue of lack of real-world training data, we train our model exclusively on synthetic data with controlled 3D changes. In order to permit sim2real, we keep the visual backbone frozen throughout training, and decode *difference* of registered features.

Since the notion of change necessitates a pair of corresponding regions with some differences, our formulation requires model predictions in *both* the images. For instance, if a car is present in one image but missing in the other, we expect the model to put bounding boxes around both – where the car is, and where the car “should have been”. Furthermore, since the changes in “in the wild” images are customarily **open-set**, our model is designed to be **class-agnostic**. We demonstrate that a model trained in this fashion zero-shot generalises to a wide variety of datasets including 2D scenes, 3D scenes, synthetic and real-world images.

In the following we provide an overview of the existing literature (Sec. 2), details of the proposed method (Sec. 3), experiments and results (Sec. 4), and finally some concluding remarks (Sec. 5). We will release the code, datasets and trained model.

2. Related Work

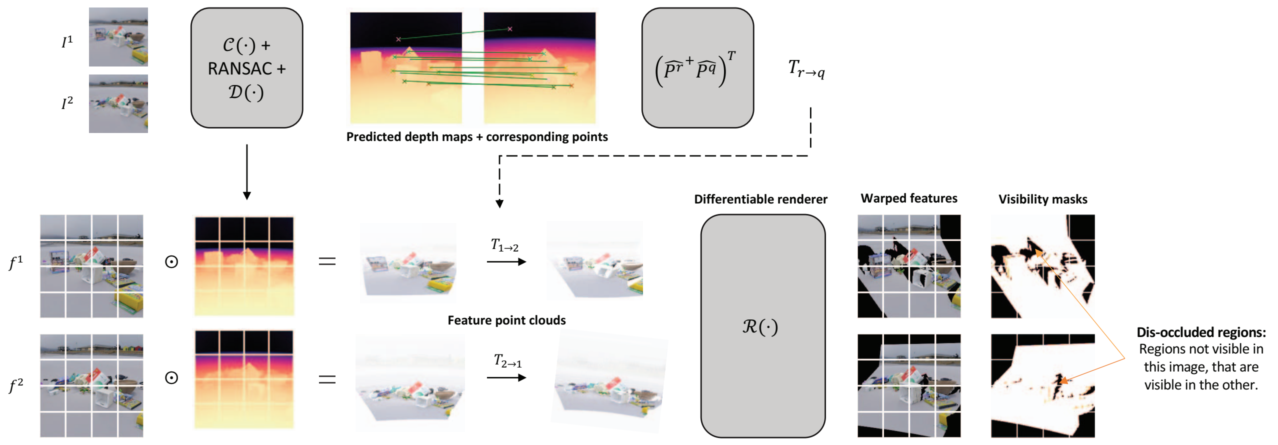
The problem of exploring visual changes has been studied in several different flavours previously. In this section we loosely group these works into two categories, 2D and 3D, and describe how our problem setting relates/differs from them.

2D (-ish): A typical scenario for the change detection problem is one where we have a pair of before-after RGB images, where the camera is either fixed i.e. the images are related by an identity transformation (e.g. images from surveillance cameras), there is a planar-scene assumption (e.g. bird’s eye view or satellite images), or in the general case there is limited shift in viewpoint (e.g. street scenes looking at distant objects/buildings), and the model is expected to identify the changes between them. Most of the existing works in the change detection literature belong to this category.

[18, 17, 21, 12] tackle the change captioning problem where the goal is to describe the changes in an image pair in natural language. These methods mainly evaluate their approach on the STD [10] (images from fixed video surveillance camera), or CLEVR-based change datasets [18, 21, 12] (synthetic images of 3D objects of primitive shapes). Since the problem these works address is change captioning, their approaches do not deal with precise localisation of changed regions.

[27, 1, 28, 14] tackle the change localisation problem, specifically for street-view images where the goal is to produce segmentation masks for changed regions. These methods mainly evaluate their approach on TSUNAMI [27] (panaromic, street-view images), VL-CMU [1] (images of urban scenes with macroscopic changes) and PCSD [28] (panoramic, street-view images). While these datasets do not technically have the planar-scene assumption or a fixed camera, the scenes are of distant buildings and there is no “peeking behind” objects due to a shift in camera pose. Since pixel-wise change annotation is expensive, these datasets often suffer from *non-comprehensively labelled test sets, limited to a set classes*.

Recently, [26] proposed a *class-agnostic* method that tackles the change detection problem for arbitrary images that are related by a homography transformation, as a bounding-box based detection problem. They evaluate their approach on STD [16], Kubric-Change [26] (3D objects resting on a 2D plane, camera is not fixed but the images



I^1
I^2
$\mathcal{C}(\cdot)$ + RANSAC + $\mathcal{D}(\cdot)$
Predicted depth maps + corresponding points
$\left(\widehat{P^r}^+ \widehat{P^q}\right)^T$
$T_{r\to q}$
f^1
f^2
$T_{1\to 2}$
$T_{2\to 1}$
Feature point clouds
Differentiable renderer
$\mathcal{R}(\cdot)$
Warped features
Visibility masks
Dis-occluded regions: Regions not visible in this image, that are visible in the other.

Figure 2. **DFRM:** Given two images, we first obtain corresponding points and depth maps which are used to estimate the 3D transformation $T_{r\to q}$. Then given a feature map for each image along with the depth map, we create a point cloud of c-dimensional feature vectors, warp it using the estimated transformation and render it to 2D grid. Notice the black regions in the rendered grid. These regions are dis-occluded and contain c-dimensional **0** vectors. We obtain a soft visibility mask to counteract the effect of dis-occluded regions when computing the difference.

are captured from bird's eye view) and others.

Similar to the methods above, we also tackle the change detection problem in a pair of RGB images. However, unlike previous methods our setting involves **general two-view images of 3D scenes** (we do not assume fixed camera or planar scenes and in our case there is a significant shift in camera pose) and we particularly focus on making our model work on **open-set**, real-world images. The problem formulation closest to ours is that of [26] in that [26] also train a class-agnostic model that produces bounding boxes around changed regions in both the images except they train and evaluate on 2D scenes.

3D: The change detection problem has also been studied explicitly for 3D data (e.g. multiview before-after images, 3D scans etc.). [19] tackle the change captioning problem in a 3D setting by assuming multi-view images are available both before and after the change. [20] further propose an end-to-end framework for describing scene changes from various input modalities, namely, RGB images, depth images, and point cloud data. Recently, [22] proposed a task to explicitly localise changes in the form of 3D bounding boxes from two point clouds and describe detailed scene changes *for a fixed classes of objects*. Unlike these works, we only assume access to a single-view image for both before and after scenes and **do not operate on explicit 3D data** like point clouds. Since RGB images are more readily available, it allows our method to be easily applied to real-world scenes. The closest setting to our is of [5] who tackle the change detection problem for general two-view images, except their model is trained and tested on the same synthetic setting with 4 fixed classes[1].

3. Method

3.1. Overview

Given an image pair of a 3D scene, captured with a significant shift in camera viewpoints, our goal is to localise the changes between them in the form of bounding box predictions for *each* image. In particular, we only wish to capture changes in the regions that are visible in both the images while disregarding areas that appear or disappear from view due to the shift in camera pose.

Our approach, which is overviewed in Figure 2, begins by extracting dense spatial image descriptors from each image using a pre-trained transformer-based visual backbone, which are then processed to obtain feature maps at multiple spatial resolutions using a U-Net [24] style encoder. Next, in order to reason between changes and occlusions/dis-occlusions, we need 3D information. Inspired by [31], our DFRM "lifts" the features to 3D, and differentiably *registers* and renders them. After obtaining registered features, we compute their *difference* in order to identify what has changed. Finally, we decode the difference of registered features using a U-Net style decoder, followed by a bounding box prediction head. We next describe the architecture, which is illustrated in Figure 3.

3.2. Architecture

Backbone: Given two images $I^1 \in \mathbb{R}^{3\times H\times W}$ and $I^2 \in \mathbb{R}^{3\times H\times W}$, we first encode I^1, I^2 using a pre-trained frozen visual backbone, represented by $\Phi_B(\cdot)$, to obtain

[1]Not available publicly at the time of writing.

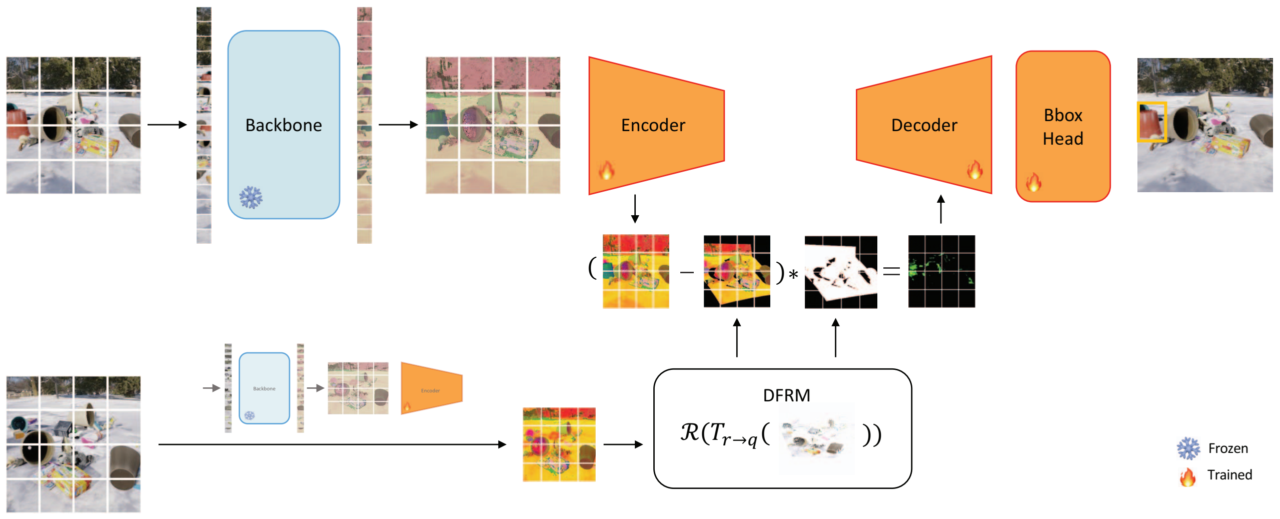


Figure 3. **Architecture:** Given two images, we first extract dense spatial feature maps using a pre-trained visual backbone. Following this, a CNN-based encoder is used to extract visual descriptors at multiple spatial resolutions. We then use a differentiable feature registration module (DFRM) to warp features from one image to another (such that they are registered), and take their difference (only one resolution is shown for the purpose of visualisation, however this operation is performed at multiple resolutions). Finally, these difference of feature maps are processed by a CNN-based decoder followed by a bounding box detection head. For brevity, we only show the prediction for one of the images, however, the pipeline is symmetric for the other. Please see Sec. 3 for more details.

two sets of feature descriptors per image represented by $f_s^1, f_d^1 = \Phi_B(I_1)$ and $f_s^2, f_d^2 = \Phi_B(I_2)$. In practise, $\Phi_B(\cdot)$ is the ViT-B/8 [6] architecture, where f_s and f_d represent shallow and deep features.

U-Net Encoder: We process f_d^1, f_d^2 using a U-Net style encoder, represented by $\Phi_E(\cdot)$, to obtain $g_n^1 = \Phi_E(f_d^1)_n$ and $g_n^2 = \Phi_E(f_d^2)_n$ after each downsampling block n, resulting in a set of multi-resolution feature maps $G^1 = \{f_d^1, g_1^1, g_2^1, g_3^1, g_4^1\}$ and $G^2 = \{f_d^2, g_1^2, g_2^2, g_3^2, g_4^2\}$, for image I^1 and I^2 respectively.

Feature Registration and Difference: Given G^1, G^2, we use a differentiable feature registration module (described in Sec. 3.3) to obtain a *warp* of feature maps at each spatial resolution such that the original features of one image are registered with the warped features of the other image. We then compute their element-wise difference and mask out occluded/dis-occluded regions. Specifically, for feature maps at spatial resolution i, we obtain the feature map $H_i^1 = \upsilon_{2\to1}(G_i^1 - \tau_{2\to1}(G_i^2))$ and $H_i^2 = \upsilon_{1\to2}(G_i^2 - \tau_{1\to2}(G_i^1))$, for I^1, I^2 respectively, where $\tau_{r\to q}$ represents the 3D feature warp operator from image I^r to I^q, and $\upsilon_{r\to q}$ represents the *soft* visibility mask.

U-Net Decoder: Following this, we decode the set of feature maps H^1 and H^2 using a U-Net decoder modulated with scSE blocks [25], represented by $\Phi_D(\cdot)$ to produce feature maps k^1 and k^2 respectively.

Bbox Head: Finally, feature maps $[f_s^1 \parallel k^1]$ and $[f_s^2 \parallel k^2]$, where $[\ \parallel\ ]$ is the concatenation operation (along channel dimension), are fed into a CenterNet head [32], which minimises the detection loss function as described in [32], to produce bounding boxes around changed regions in both the images. The motivation for concatenating shallow features f_s with k is that it serves as the final skip connection before the prediction head, which is consistent with the typical U-Net style model, and additionally shallow features are known to capture more *positional* information, as opposed to deep features which capture more *semantic* information [2], and therefore can help with localisation.

3.3. Differentiable Feature Registration Module (DFRM)

Given images $I^r, I^q \in \mathbb{R}^{3\times H\times W}$, along with their feature maps $f^r, f^q \in \mathbb{R}^{c\times h\times w}$ respectively, we use the following three-step process, represented by $\tau_{r\to q}(\cdot)$, to warp f^r such that f^q and $\tau_{r\to q}(f^r)$ are registered. See Figure 2 to visually conceptualise all the moving parts.

Step 1: Estimate a 3D linear transformation. In order to register the two feature maps, we must estimate a 3D transformation between the two images. To do so, first we obtain a set of n corresponding points $P^r, P^q \in \mathbb{R}^{n\times 2}$ (in normalised coordinates) in image I^r, I^q respectively, using a correspondence extractor represented by $\mathcal{C}(\cdot)$. Following this, we estimate the depth maps $D^r, D^q \in \mathbb{R}^{3\times H\times W}$ using a monocular depth estimator represented by $\mathcal{D}(\cdot)$. Finally, to estimate the transformation, we first back-project each point $(x_j, y_j) \in P^r, P^q$ as,

$$\hat{P}_j = \begin{bmatrix} d_j x_j & d_j y_j & d_j & 1 \end{bmatrix}^T \tag{1}$$

where d_j is the depth value at point P_j, resulting in two sparse 3D point clouds $\hat{P^r}, \hat{P^q} \in \mathbb{R}^{n\times 4}$ in homogenous coordinates. Following this, we estimate a transformation matrix $T_{r\rightarrow q} \in \mathbb{R}^{4\times 4}$, such that $T_{r\rightarrow q}\hat{P}^r_j \approx \hat{P}^q_j$, using the following closed-form solution:

$$T_{r\rightarrow q} = \left(\hat{P^r}^+ \hat{P^q}\right)^T \tag{2}$$

where $\hat{P^r}^+ \in \mathbb{R}^{4\times n}$ is the Moore-Penrose inverse of $\hat{P^r}$.

Step 2: Lift the features to 3D and warp. For each c-dimensional feature vector in $f^r \in \mathbb{R}^{c\times h\times w}$, we project its normalised 2D grid coordinates (x_j, y_j) to a point $\left(x^{'}_j/k,\, y^{'}_j/k,\, z^{'}_j/k \right)$ in 3D, where

$$\begin{bmatrix} x^{'}_j & y^{'}_j & z^{'}_j & k \end{bmatrix}^T = T_{r\rightarrow q} \begin{bmatrix} d_j x_j & d_j y_j & d_j & 1 \end{bmatrix}^T \tag{3}$$

where d_j is the estimated depth value of this point, resulting in a 3D point cloud of f^r feature vectors that are aligned with f^q.

Step 3: Differentiable feature rendering. Given the 3D point cloud, we render it to the 2D grid using a differentiable renderer $\mathcal{R}(\cdot)$. However, instead of rendering RGB colours, we render c-dimensional feature vectors for each point. For this, we employ a differentiable point cloud renderer, as in [31]. The advantages of this differentiable point cloud renderer are two fold: (1) it solves the "small neighbourhood" problem, wherein each feature-point projects to only one or a few pixels in the rendered view, by splatting the points to a disk of controllable size, and (2) the "hard z-buffer" problem, wherein each rendered pixel is only affected by the nearest point in the z-buffer, by accumulating the effects of K nearest points. Both of these allow for better gradient propagation during training. In addition to obtaining the rendered features, we also obtain a visibility mask $v_{r\rightarrow q}$ to deal with occluded/dis-occluded regions (see Figure 2). This is obtained by setting the 2D coordinates of the rendered points to 1 on a grid initialised with 0s. However, due to the splatting behaviour of the renderer, the visibility mask obtained is *soft* and not binary.

Alternate registration strategies: Since DFRM is training-parameter free, during inference $\tau_{r\rightarrow q}(\cdot)$ may utilise alternate registration strategies. For instance, if ground truth depth is known for each image, it can directly replace D^r, D^q obtained using the monocular depth estimator $\mathcal{D}(\cdot)$. In addition to ground truth depth, if the camera intrinsics and extrinsics are known, points can be directly back-projected and warped using relative camera poses without needing to estimate the transformation $T_{r\rightarrow q}$. Furthermore, if only D^r is available, Perspective-n-Point methods [9] can be used to warp features f^r onto f^q.

On the other hand, if it is known apriori that the images are of planar scenes, depth is not needed and each 2D grid coordinate can be warped using $T_{r\rightarrow q} \in \mathbb{R}^{3\times 3}$ which can either be supplied or estimated using standard homography estimation methods. If the scene consists of multiple planes (e.g. floor and wall), it may also be possible to obtain desired results using a multi-grid [15] or multi-plane homography estimation [9] to register the images.

3.4. Details and discussion

The ViT-B/8 [6] backbone $\phi_B(\cdot)$ is initialised with DINO model weights [4]. DINO features are known to encode powerful high level semantic information at fine spatial granularity and recent works have shown that scalar product of DINO features can be used to compute high quality semantic correspondences [2]. Orthogonally to computing correspondences, we utilise these powerful DINO features to compute changes. To prevent the model from overfitting to synthetic data and corrupting the quality of DINO features, we keep the backbone frozen during training. In order to allow sim2real, our decoder only ever operates on the *difference* of features, which is negatively proportional to the scalar product i.e. the higher the scalar product the smaller the difference, and therefore we only consider the similarity of features (or the lack of) rather than the features themselves (whether synthetic or real). Following [2], we increase the resolution of model features by changing the stride of the patch extraction (from 8 to 4) and adjusting the positional encoding appropriately via interpolation, allowing us to operate on more granular spatial features. The shallow features and deep features f_s, f_d are extracted from the *keys* of the Multiheaded Self-Attention layers of the third and the last block respectively, an insight also from [2].

For correspondence extractor $\mathcal{C}(\cdot)$, we experimented with [11, 30] but found SuperGlue [29] to work the best at generating high quality dense correspondences and at high inference speed. After extracting 2D correspondences from SuperGlue, we filter out outliers using RANSAC. Finally, for the monocular depth estimator $\mathcal{D}(\cdot)$, we tried [23, 7] but found the recently released ZoeDepth [3] to consistently produce better results.

4. Experiments

This section describes the data we used to train our model, evaluation benchmarks and baselines, and various implementation details. Please see Table 1 for an overview of the datasets used and Figure 4 for some example images.

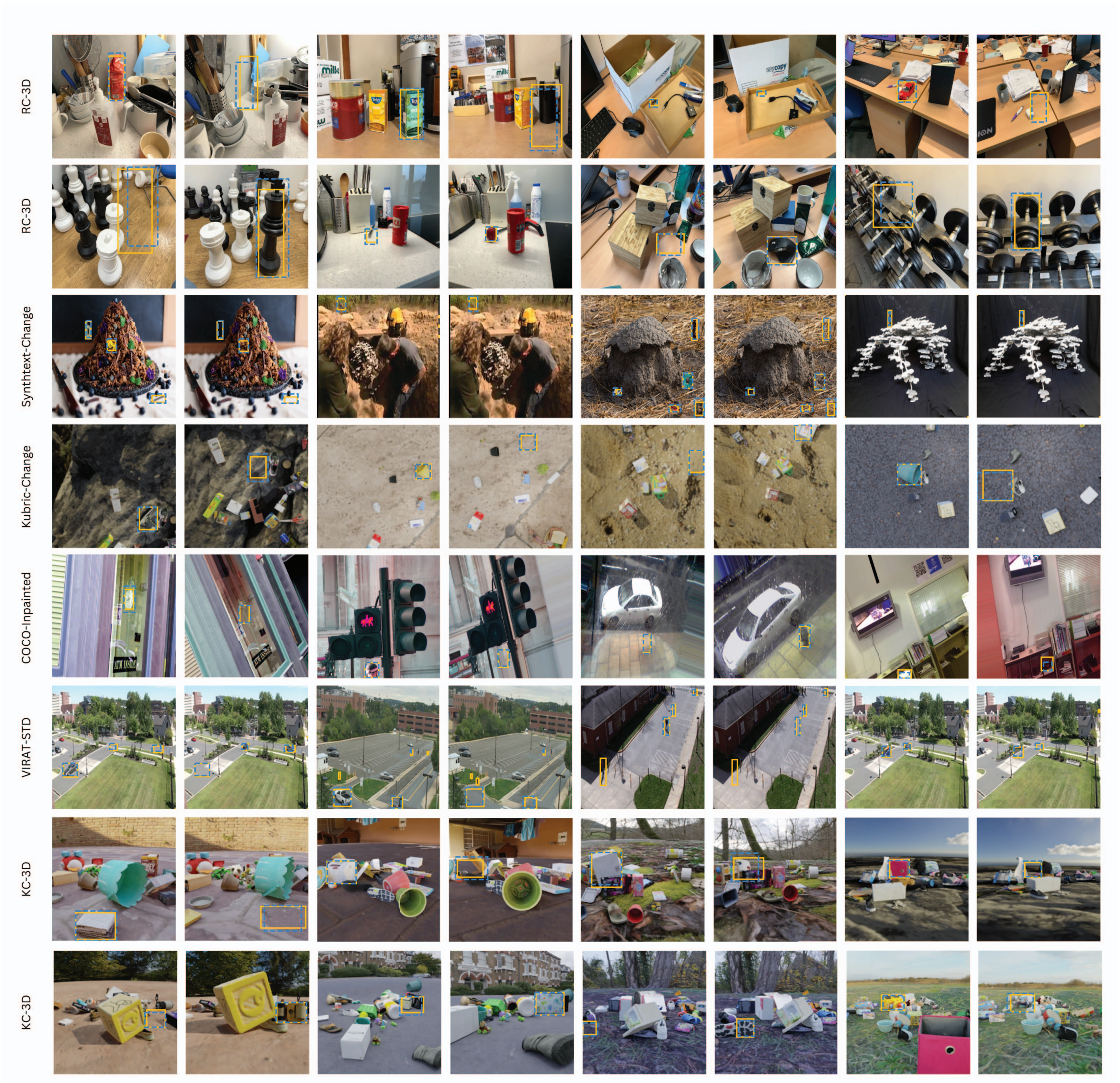


Figure 4. **Qualitative results:** We show the bounding box predictions in yellow (solid) of our model on all the test sets, along with the ground truth in blue (dashed).

4.1. Training datasets

Due to the lack of existing large-scale real-world datasets for the change detection problem, as formulated in this work, we fall-back to training our model entirely on synthetic data. Specifically, we train our model jointly on the following two datasets:

KC-3D: Similar to [26], we make use of the Kubric dataset generator [8] to curate 86407 image pairs (of which 4548 images pairs are for validation) of 3D scenes with controlled changes. The scenes consist of randomly selected set of 3D objects spawned at random locations on a randomly textured plane, where we iteratively remove the objects and capture "before" and "after" image pairs. However, unlike [26] where they capture birds-eye view images only, we capture images from a wide variety of

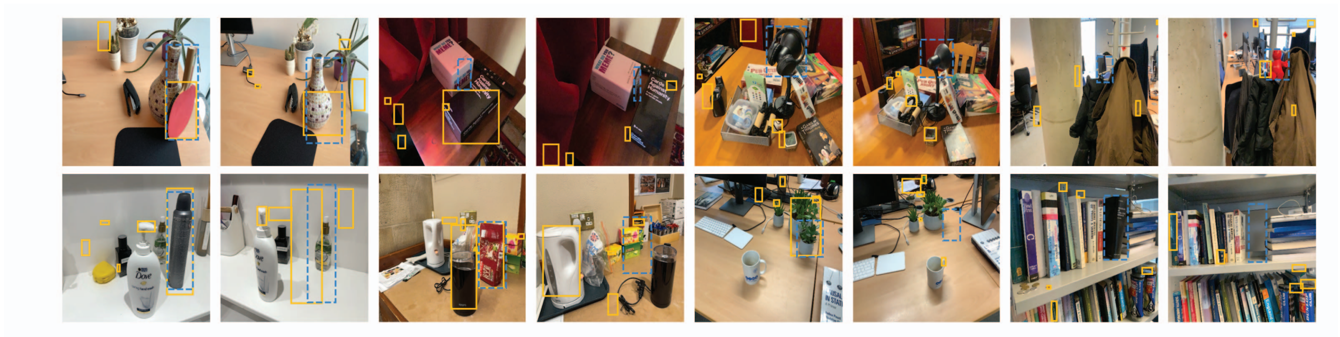


Figure 5. **Failure cases:** Here we show some RC-3D images where our model failed to localise the correct changed region. We classify an output as a *failure* if the top-5 most confidence bounding boxes do not contain the ground truth changed region. Predictions are in yellow (solid), ground truth is in blue (dashed).

camera poses inside a cylindrical space around the objects. In addition, we also capture the depth maps, camera intrinsics and extrinsics for each image to supply to DFRM during training.

COCO-Inpainted: While KC-3D captures the underlying challenges of our task in terms of viewpoint shift quite well, it lacks diversity in terms of kinds and sizes of objects. Therefore, we additionally utilise the recently introduced COCO-Inpainted dataset [26] for training. While this is a 2D dataset, where the image pairs are perturbed by an affine transformation, it helps the model learn to predict changes of various kinds and sizes.

4.2. Testing datasets

To test the performance of our model, we evaluate it on the following test sets.

KC-3D: Following the same pipeline as before, we curate an additional 4548 image pairs of 3D scenes from Kubric for testing purposes (making the total number of image pairs in KC-3D to be 90955).

RC-3D: To quantify our model's capacity to generalise to real-world images, we manually collect and label a small-scale test set consisting of 100 images pairs, capturing a diverse set of common objects found in everyday places like office, kitchen, lounge etc. The images were captured using a handheld Apple iPad Pro (4th Gen) and Apple iPhone 14 Pro, which come with a built-in LiDAR giving us aligned RGB-D images.

Cyws Test Sets: Since the problem we are tackling subsumes the change detection problem in planar scenes, we also evaluate our model on 2D test datasets proposed in [26], namely COCO-Inpainted, VIRAT-STD, Kubric-Change, and Synthtext-Change.

Test set	COCO-Inpainted	Synthtext-Change	VIRAT-STD	Kubric-Change	KC-3D	RC-3D
type	inpainting	text	surveillance	sim	sim	general 2-view
change	synthetic	synthetic	real	synthetic	synthetic	real
geometry	affine	identity	identity	2D-ish (bird's eye)	3D	3D
# train images	60000+	-	-	-	81859	-
# test images	4408	5000	1000	1605	4548	100

Table 1. **Datasets.** KC-3D and RC-3D are ours. Others are from [26].

4.3. Baseline and metrics

To the best of our knowledge, no prior works have tackled the change detection problem in a *general two-view and class-agnostic* setting like us which makes it difficult to directly compare our work with prior art. Nevertheless, we use cyws [26] as a baseline as their formulation is the same as ours, except restricted to 2D transformations. To allow for a fair comparison, in addition to reporting the results using their open-sourced model, we also finetune their pretrained model on the same training dataset as ours for 100 epochs (best model is picked using lowest loss on val set). We use the average precision metric to report our results, similarly to [26].

4.4. Training details

We trained the model on 4$\times$ A40 GPUs for 50 epochs using the DDP strategy with a batch size of 16, where the best model is chosen using a validation set (COCO-Inpainted val set as in [26] + a KC-3D val set). Images were augmented with CropAndResize, HorizontalFlips and ColourJittering, and resized to 224 $\times$ 224 due to the DINO-ViT [4] requirements. Since our training data is entirely synthetic, during training we use ground truth data for $\tau_{r\to q}(\cdot)$ rather than estimating it. The overall objective was optimised using Adam [13] with learning rate of 0.0001 and weight decay of 0.0005.

4.5. Results

We evaluate our model, which is trained on synthetic data described in Sec. 4.1, on a diverse set of testing datasets as described in Sec. 4.2 with no further training/finetuning.

	test dataset	COCO-Inpainted small	medium	large	all		VIRAT-STD	Synthtext-Change	Kubric-Change	KC-3D		RC-3D	
	depth	Const.					Const.	Const.	Const.	GT		GT	Est.
	registration	GT			GT	Est.	Id.	Id.	Est.	GT	Est.	Est.	
method	training data	(Const. = Constant, Id. = identity, Est. = Estimated, GT = Ground Truth)											
cyws	coco-inpainted	**0.46**	**0.79**	**0.85**	**0.63**		**0.65**	**0.89**	0.76	0.13		0.12	
cyws	coco-inpainted + KC-3D	0.41	0.73	0.78	0.57		0.54	0.87	0.76	**0.87**		0.14	
ours	coco-inpainted	0.34	0.69	0.76	0.52	0.51	0.46	0.85	**0.84**	0.14	0.10	0.35	0.27
ours	KC-3D	0	0.03	0.06	0.02	0.02	0.01	0.01	0.23	0.83	0.69	0.19	0.19
ours	coco-inpainted + KC-3D	0.36	0.72	0.77	0.53	0.52	0.49	0.84	**0.84**	0.82	0.68	**0.50**	0.41

Table 2. **Results:** We report the AP of cyws [26] and our model on test sets described in Sec 4.2.

Table 2 contains the quantitative results in terms of average precision, while we show some qualitative predictions of our model in Figure 4 and some failure cases in Figure 5.

3D scenes (KC-3D, RC-3D): From Table 2 it is evident that our model produces impressive results on both synthetic and "in the wild" real-world images. Particularly in the case of RC-3D, the model produces almost 4$\times$ better results than the baseline and performs well even in the most challenging setting when only RGB image pairs are available as input. Despite only having been trained on synthetic data, the remarkable performance of the model on real-world images validates the design choice of only using feature differences and not features directly (unlike cyws, where their co-attention module concatenates the cross-attended features). On the other hand, we observe a surprising result from the cyws model, which is able to produce impressive results on KC-3D dataset when finetuned on it. It is likely that the cyws model is "over-fitting" to the Kubric setting given that the set of objects and scenes in Kubric [8] are limited (different scenes just have different random combinations from the same set of objects and backgrounds). Despite this fact, it is still interesting that the model is able to reason about changes despite not being 3D aware. Nevertheless, this performance does not generalise to real-world images as observed by its poor results on RC-3D.

2D/fixed-camera scenes (Cyws test sets): In the 2D setting, we found that our model is often comparable but not strictly better than cyws. In particular, we observed that our model particularly struggles with detecting really small changes in comparison to cyws which is likely due to the fact that cyws operates at higher input resolution than us (256×256 vs 224×224) and that the number of trainable parameters in our model is 31.5M which is much less than 49.5M in cyws. Furthermore, we found that a lot of the annotations (for COCO-Inpainted small and VIRAT-STD), are extremely small (handful of pixels, almost indiscernible to human-eye). This becomes problematic for our model when the input resolution is reduced to 224×224. In addition, the ground-truth annotations for VIRAT-STD are noisy (both false positives and missing annotations, see Figure 4 where our model predicts valid changes that are missing from ground truth).

4.6. Limitations

Despite the remarkable ability of our model to localise changes in real-world general two-view images, it suffers from a few limitations. A potential concern is the large size of the model. While the number of trainable parameters is 31.5M (which is less than 49.5M in cyws), accounting for the frozen DINO-ViT backbone [4] with 85.8M parameters, our total model size is roughly 117M parameters. This makes it much slower to train and infer from than cyws. However, it must be acknowledged that this large backbone comes with the added benefit of making our model generalisable to real-world images even though it has only been trained on synthetic data. Another potential cause of concern is that it relies on a good estimated registration, which in turn relies on reliable correspondences and depth, which may not be always available. On the other hand, it also means that as better correspondence extractors and monocular depth estimators become available, our model's results can improve at no additional cost.

5. Conclusion

In the ever-changing landscape of our world, the task of detecting changes in a 3D scene is daunting for both humans and machines alike. In this work we take a step closer towards solving this problem by automatically detecting changes in real-world images captured from significantly different viewpoints. Due to the lack of large-scale real-world training datasets for this problem, we propose a model that is trained entirely on synthetic data but can generalise to real-world scenes by leveraging the recent advances in computer vision. Following previous works, we largely focused on detecting missing objects as they are easy to acquire and precise. However, we note that our model should not be confused with a "(missing) object detector". While it is trained on object-centric datasets, our model can zero-shot detect all sorts of open-set changes.

References

[1] Pablo F. Alcantarilla, Simon Stent, German Ros, Roberto Arroyo, and Riccardo Gherardi. Street-view change detection with deconvolutional networks. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016. 2

[2] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ECCVW What is Motion For?*, 2022. 4, 5

[3] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth, 2023. 5

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 5, 7, 8

[5] Kento Doi, Ryuhei Hamaguchi, Yusuke Iwasawa, Masaki Onishi, Yutaka Matsuo, and Ken Sakurada. Detecting object-level scene changes in images with viewpoint differences using graph matching. *Remote Sensing*, 14(17):4225, 2022. 3

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4, 5

[7] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10786–10796, 2021. 5

[8] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3749–3761, 2022. 6, 8

[9] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 5

[10] Harsh Jhamtani and Taylor Berg-Kirkpatrick. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. 2

[11] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. Cotr: Correspondence transformer for matching across images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6207–6217, 2021. 5

[12] Hoeseong Kim, Jongseok Kim, Hyungseok Lee, Hyunsung Park, and Gunhee Kim. Agnostic change captioning with cycle consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2095–2104, 2021. 2

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[14] Yinjie Lei, Duo Peng, Pingping Zhang, Qiuhong Ke, and Haifeng Li. Hierarchical paired channel fusion network for street scene change detection. *IEEE Transactions on Image Processing*, 30:55–67, 2021. 2

[15] L Nie, C Lin, K Liao, S Liu, and Y Zhao. Depth-aware multi-grid deep homography estimation with contextual correlation. arxiv 2021. *arXiv preprint arXiv:2107.02524*. 5

[16] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *CVPR 2011*, pages 3153–3160. IEEE, 2011. 2

[17] Ariyo Oluwasanmi, Enoch Frimpong, Muhammad Umar Aftab, Edward Y. Baagyere, Zhiguang Qin, and Kifayat Ullah. Fully convolutional captionnet: Siamese difference captioning attention model. *IEEE Access*, 7:175929–175939, 2019. 2

[18] Dong Huk Park, Trevor Darrell, and Anna Rohrbach. Robust change captioning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4623–4632, 2019. 2

[19] Yue Qiu, Yutaka Satoh, Ryota Suzuki, Kenji Iwata, and Hirokatsu Kataoka. 3d-aware scene change captioning from multiview images. *IEEE Robotics and Automation Letters*, 5(3):4743–4750, 2020. 3

[20] Yue Qiu, Yutaka Satoh, Ryota Suzuki, Kenji Iwata, and Hirokatsu Kataoka. Indoor scene change captioning based on multimodality data. *Sensors*, 20(17):4761, 2020. 3

[21] Yue Qiu, Shintaro Yamamoto, Kodai Nakashima, Ryota Suzuki, Kenji Iwata, Hirokatsu Kataoka, and Yutaka Satoh. Describing and localizing multiple changes with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1971–1980, 2021. 2

[22] Yue Qiu, Shintaro Yamamoto, Ryosuke Yamada, Ryota Suzuki, Hirokatsu Kataoka, Kenji Iwata, and Yutaka Satoh. 3d change localization and captioning from dynamic scans of indoor scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1176–1185, 2023. 3

[23] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 5

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[25] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks. In *International conference on medical image computing and computer-assisted intervention*, pages 421–429. Springer, 2018. 4

[26] Ragav Sachdeva and Andrew Zisserman. The change you want to see. In *Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2, 3, 6, 7, 8

[27] Ken Sakurada and Takayuki Okatani. Change detection from a street image pair using cnn features and superpixel segmentation. pages 61.1–61.12, 01 2015. 2

[28] Ken Sakurada, Mikiya Shibuya, and Weimin Wang. Weakly supervised silhouette-based semantic scene change detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6861–6867, 2020. 2

[29] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 5

[30] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021. 5

[31] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 3, 5

[32] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 4