Traffic Sign Substitution

Exploring effective ways to substitute pre-existing traffic signs in images with other signs.

Sayan Mandal
Mat.Nr. 12131708

Institute of Computer Graphics and Vision
Graz University of Technology, Austria

Seminar/Project Computer Vision
Prof. Horst Bischof
Graz, May 25, 2023

Abstract

Traffic sign detection and recognition is a key yet challenging task for autonomous driving systems due to a large number of classes and natural conditions in the wild. Dataset gathering for this task is both costly and difficult given the diversity of classes and appearances depending on country of origin and weather condition of the time of capture.
In this work, we will focus on creating realistic and aesthetic traffic sign composites for object detection and instance segmentation, given a source traffic sign object, it's own straight prototype and a target straight prototype. For substitution, we follow a homography based approach by finding the desired homography matrix between the source sign and it's own prototype and using that matrix to transform the target prototype. In overall, the project follows a four step process: 1) use inpainting to remove foreground traffic sign A in original image, 2) use homography to transform prototype sign B to match A, 3) blend transformed sign B in A's location 4) use harmonization to achieve visual consistency in the composite location. The pipeline can be used as a data augmentation technique to both train new and test existing instance segmentation and detection models. In order to evaluate overall substitution performance, we use ResNet50 based Mask R-CNN trained on original images and tested on original images vs substitution images. The code is available at: https://github.com/smandal94/trafficsign_substitution.git

Keywords: *Traffic sign, homography, image inpainting, image harmonization, substitution*

1 Introduction

Traffic sign detection and recognition are important tasks for both Advanced Driver Assistance Systems (ADAS) and autonomous vehicles. These tasks are challenging problem in training and testing as:

- the possible number of classes are extensive and differs from country to country
- the variance in appearance due to weather, background and lighting conditions
- the data gathering cost incurred
- unbalanced class distribution

In order to mitigate the above problems, the natural course of action is to create a standard traffic sign detection and recognition dataset of a specific country and then use traffic sign substitution to create a large-scale augmented dataset. Given the target object placement coordinates in an image via instance segmentation masks and bounding box coordinates, direct composition involves cutting the foreground from a different image or using template foreground image and pasting it on the target placement coordinates but this method suffers from geometric and appearance inconsistency in the foreground and the background. Appearance inconsistencies refer to i) unnatural boundary between foreground and background, ii) incompatible color and illumination statistics between foreground and background and iii) missing or implausible shadow and reflection of foreground. Geometric inconsistencies include i) foreground object being out of shape and ii) inconsistent perspective between foreground and background etc. In order to mitigate the above mentioned inconsistencies, our substitution pipeline follows a 4-step process as mentioned below:

Given a source image I_{org}, its instance segmentation map of k^{th} traffic sign M_k, k^{th} template T_k and a target template sign T_t,

1. Object Removal of k^{th} sign in I_{org} using Image Inpainting to get $I_{inpaint}$
2. Object Placement of T_t in the location of k^{th} sign in I_{org} with geometric consistency, via projective transformation using homography matrix $H\_T_k\_k$
3. Image blending of transformed T_t on $I_{inpaint}$ without visual artifacts and jagged edges
4. Image Harmonization of T_t in $I_{inpaint}$ w.r.t $I_{inpaint}$ background for appearance consistency to get result image I_r

The advantage of this 4-step pipeline is that the steps are research fields on their own and we can use SOTA pretrained networks of the respective fields in our pipeline as Plug-and-Play modules.

In this work we conduct different experiments on instance segmentation based DFG Traffic Sign Dataset [35] and select LaMa [34] for step 1, SuperGlue+SuperPoint [9, 28] for calculating homography in step 2, alpha blending in step 3 and RainNet [19] for image harmonization in step 4. To judge the overall substitution performance, we first conducted visual checks between original and substituted images and then evaluate the instance segmentation performance of Mask R-CNN [12] with ResNet50 [13] backend (pretrained on MS-COCO dataset [18]), which was trained on original images and tested on the original testset and substitution testset. In the substitution testset, all original traffic signs are replaced with other randomly picked traffic signs.

2 Related Works

Traffic sign substitution has been tried out for augmenting sign classification dataset in [15] and [32].

In [32], the authors proposed a CycleGAN [44] based style transfer technique between icon-like arbitrary traffic signs to life-like images of traffic signs. They transfer prototype icon-like image with random homogeneous background color to life-like image with varied scenic details. The model learns an association between the homogeneous background colors and certain realistic background styles. The model was trained and evaluated on GTSRB dataset [33].

Extending the above work in [15], the authors noted that [32] resulted in decrease in classification rate by 5 to 20 percentage, when training a classifier on a comparable number of generated traffic signs instead of their real counterparts. They attributed this drop in performance to the low-quality backgrounds in the generated images. Firstly, the generated backgrounds lacked consistency for larger structures like buildings etc. and seemed to prefer vegetation. Secondly, mountings like poles etc. to which the signs are attached to, are useful for the traffic sign classifier's inference but they were neglected or omitted by their said previously proposed generation process. In [15], they addressed these shortcomings by replacing traffic signs in real images by artificially created ones. They created a two-stage pipeline: extraction and composition. In extraction, they used CycleGAN on real-world samples from GTSRB, to create cartoon pendant from which they extract a binary background segmentation map and estimates the traffic sign pose via matching ORB features [27]. In composition, they take a substitute icon and apply inverse of pose calculated earlier to get the tilted version. The background is replaced by the background of cartoon generated earlier and then CycleGAN trasfers it to the life-like domain. The segmentation map in extraction phase is used to paste this generated life-like icon to the original real background with border crossfading to avoid artifacts. The generation process from paper is depicted in Fig.1.

Above works focus on traffic sign substitution in image classification dataset GTSRB. Since our focus is traffic sign substitution on full image scene, i.e datasets suited for training/testing

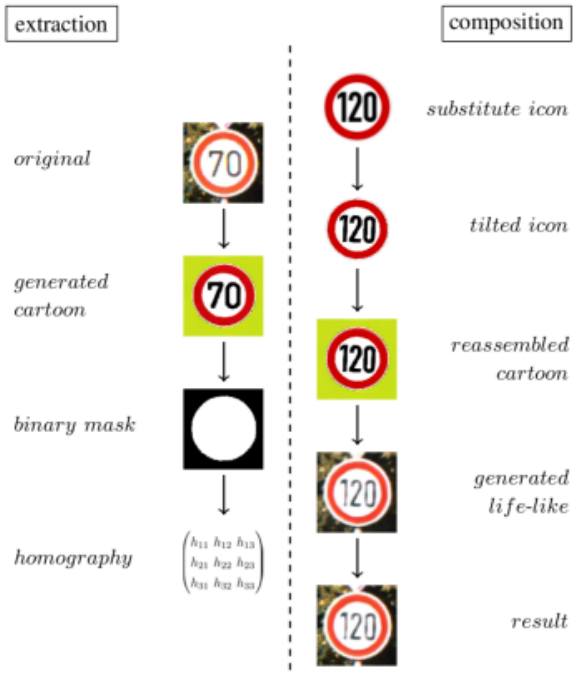


Figure 1: Image generation process from [15] paper.

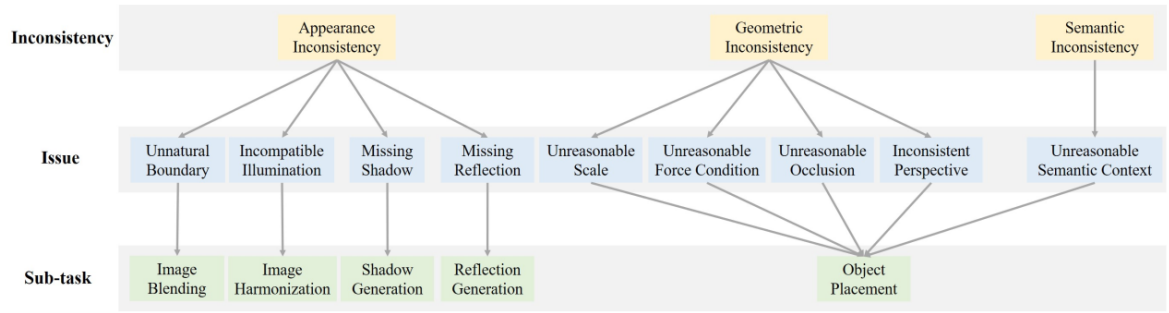


Figure 2: Image composition issues as highlighted in [22] paper.

detection and segmentation tasks, we will look into the image composition domain as a whole. To this end, [22] identified the major problem faced in image composition being appearance inconsistency, geometric inconsistency and semantic inconsistency (Fig. 2) and surveyed on different works in the individual research directions pertaining to above mentioned inconsistencies, namely: i) Object Placement, ii) Image Blending, iii) Image Harmonization, iv) Object Shadow Generation, v) Composition Aware Aesthetic Evaluation and vi) Image Manipulation Detection.

In order to prevent the pre-existing traffic sign from sticking out from below the substituted sign, we first look into image inpainting models to remove the pre-existing sign foregrounds before replacement. [36] is based on Fast Marching method for pixel traversal. The algorithm uses a small neighborhood of the pixel to be inpainted and replaces the pixel by the normalized weighted sum of the neighborhood. The pixel traversal starts from the boundary of the inpainting region and gradually moves inside the region. [4] algorithm uses fluid dynamics and partial differential equations. It continues isophotes and matches gradient vectors at the boundary of the inpainting region using methods of fluid dynamics. Because edges are meant to be continuous, it traverses along edges from known regions to unknown regions. In EdgeConnect [21], the authors proposed a two-stage adversarial model comprising of an Edge Generator which predicts the full edge map given a masked grayscale image, its edge map and the image mask and an Image Completion Network that fills in the missing regions given the full edge map as a priori and the incomplete color image. The benefit of this is the exhibition of finer details in filled regions as compared to over-smoothed and/or blurry structures in previous works. The Edge Generator uses joint adversarial and feature matching loss while the Image Completion Network uses joint l_1 loss, adversarial loss, perceptual loss and style loss. LaMa [34] addresses the failure of inpainting models to inpaint large missing areas and complex geometric structures in high resolution images by using fast Fourier convolutions [6] in the inpainting network, a high receptive perceptual loss with dilated convolutions and training the network with large masks.

Since the foregrounds can be considered as views of the same plane, we will need to calculate the homography matrix to transform one view to another. One way to calculate is to find atleast four matching keypoints on the image pair and then use DLT with least squares or RANSAC method to calculate the homography matrix. Another way is to use a network that takes the image pair as input and directly predicts the homography H_1_2. Many traditional interest point detectors like SIFT [20], SURF [3], ORB [27], KAZE [2], BRIEF [5], AKAZE [1], etc. are available for extracting robust keypoints. [16] studied the performance of SIFT, SURF and ORB under different kinds of image distortions. In their experiments, SIFT had the best matching rate under varying intensity, shearing and fisheye distortion. Under rotation, SIFT provided the best matching rate for angles 45, 135 and 225 degrees whereas ORB and SURF gave better matches for 90 and 180 degree angles. ORB provided the best matching rate under scaling and salt and pepper noise with both ORB and SIFT achieving almost equal performance under salt and pepper noise. In SuperPoint [9], the authors proposed a self-supervised framework for training

a fully-convolutional neural network for computing SIFT-like interest points and descriptors. SuperGlue [28] introduced a Graph Neural Network for matching detected keypoints in an image pair. In order to find the visual and spatial relationships between the set of keypoints, the network uses cross and self attention mechanisms. In MatchFormer [39], the authors proposed an extract-and-match transformer pipeline having interleaved self-attention for feature extraction and cross-attention for feature matching. HomographyNet [8] used a 10-layer VGG-style feed forward network to directly predict a 4-point parameterized homography. They experimented with a classification network that produces a distribution over quantized homographies and a regression network that directly estimates the real-valued homography parameters. They used Mean Average Corner Error (MACE) to evaluate the performance of their models and compare with ORB+RANSAC and the regression HomographyNet scored the lowest error.

Once the geometric consistency is achieved by homography estimation and guidance of alpha masks, it is important for the substitute foreground to achieve appearance consistency with the background and similar to the original foreground. Alpha blending [25] uses alpha masks to do image layering with the alpha values deciding what percentage of the lower layer color is visible through the current layer color. Poisson blending [23] imposes a gradient domain fusion on the composite image by reconstructing pixels in the blending region such that the composite image has small gradients or smooth transition with respect to the background boundary pixels. [41] proposed a poisson blending loss and a two-stage model where the first stage focuses on seamless blending and second stage focuses on style refinement. In [40], the authors proposed Gaussian-Poisson Generative Adversarial Network (GP-GAN) that leverages traditional gradient based approach and GAN to blend high-resolution images given their composite images. GP-GAN optimizes a loss function consisting of a color constraint and a gradient constraint. They propose Blending GAN to generate semantically realistic image from a copy-paste image and this generated image is used as the color constraint. Gaussian-Poisson equation is proposed to generate high-resolution textures and edges.

In case the composite foregrounds are precisely delineated at the background during pasting, with no jagged boundaries and color artifacts along the boundary, one can use skip the blending algorithms and go for image harmonization methods to adjust the color and illumination statistics of the composite foreground to make it more harmonious with the background. To this end, [31] proposed incorporation of high level semantic features from a pre-trained foreground aware semantic segmentation architecture into pre-existing encoder-decoder architectures used in image harmonization. In RainNet [19], the authors proposed Region-aware Adaptive Instance Normalization (RAIN) module to solve the image harmonization problem as a background-to-foreground style transfer problem, by explicitly formulating the visual style from the background and adaptively applying them to the foreground.

In order to evaluate the quality of generated composite images, metrics like Fréchet Inception Distance (FID) [14], Structural SIMilarity index (SSIM) [29], Learned Perceptual Image Patch Similarity (LPIPS) [42], etc. are used. Works like [37], InstaBoost [10], [15] and [32] have used the performance improvement of downstream tasks (for eg. image classification, object detection, instance segmentation) to evaluate the quality of composite images. In this work, we use visual checks to authenticate the naturalness of the composite images in all our modules. For evaluating foreground removal we use LPIPS and for evaluating the overall substitution, we use Mask RCNN with ResNet50 backend trained on original images and tested on original vs composite images. Compared to other works, we do not use our pipeline to augment the training data but simply test on augmented data to check the drop in accuracy vs original.

3 Method

[15] and [32] worked in image classification dataset and depends on CycleGAN for capturing pose, background and image characteristics in a generated traffic sign cartoon. An inaccurately generated traffic sign will result in both pose estimation and harmonization errors. We decided to decouple the calculation of pose and image statistics in this work. We go for a plug-and-play modular pipeline and break the overall substitution into four modules so as to both understand the efficacy and problems of each module as well as to find better alternatives later.

The overall pipeline is already mentioned in Sec. 1 and more elaborately depicted in Fig. 3. First, we remove all traffic signs that we want to substitute, using image inpainting in an offline manner and keep the original-inpainted pair ready. Then in the online task, for each traffic sign in the original image, we extract it using its instance segmentation mask, calculate the homography matrix $H\_T_k\_k$ between the traffic sign template and original and use $H\_T_k\_k$ to transform the substitution template to the original foreground pose. We use the bounding box annotation to blend the transformed substitution into the inpainted image, thus taking care of the possible geometric inconsistencies arising in substitution, mentioned in [22]. Finally once all substitutions are made, we use the substitution image and its mask to do image harmonization. We will explain each module in more detail in the following sub-sections.

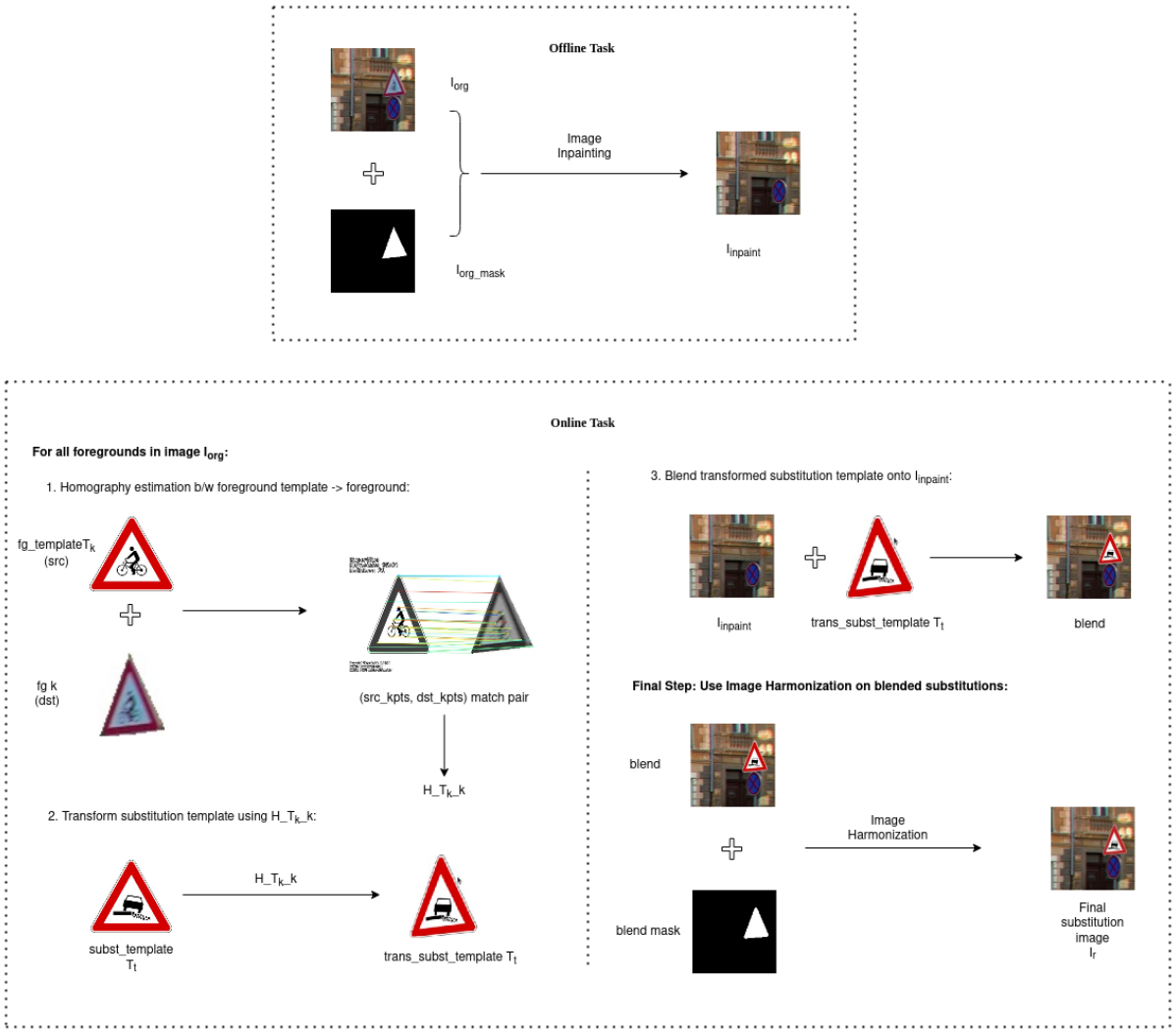


Figure 3: Our complete traffic sign substitution pipeline

Figure 4: Two samples from the DFG Traffic Sign Dataset [35] with object label, bounding box and instance segmentation mask annotations

3.1 Dataset

Our method described in Fig. 3 relies on bounding box and instance segmentation annotations of the traffic signs. We use the instance segmentation annotations to do foreground inpainting as well as to extract the traffic signs during homography estimation so as to not let the background around the traffic sign produce spurious keypoint matches. The bounding box annotations are used for object placement during substitution.

For our work, [35] suited best because it has both instance segmentation and bounding box annotations. The dataset has 6957 1920x1080 RGB images with 13239 annotations corresponding to 200 categories. The images were captured across different Slovenian municipalities by driving a camera mounted car. The captured angle is perfect for training and testing traffic sign recognition systems on autonomous vehicles. Since Slovenia is part of the Vienna Convention on Road Signs and Signals, use of this dataset makes our work applicable for all countries under that convention, including Germany and Austria. Fig. 4 shows two samples from the dataset.

For substitution, we only experiment with the testset. We exclude any traffic signs that are not over 900px area. Since matching of keypoints require the template and image sign have the same characteristics (writing/drawing), we exclude all categories that have heterogeneous content and two categories ('X-1.2', 'II-34') that lacked necessary amount of features in the foreground. Fig. 5 shows a few examples of categories having heterogeneous content that were in the exclusion list.

Figure 5: Examples of categories having heterogeneous content. All samples are from the dataset

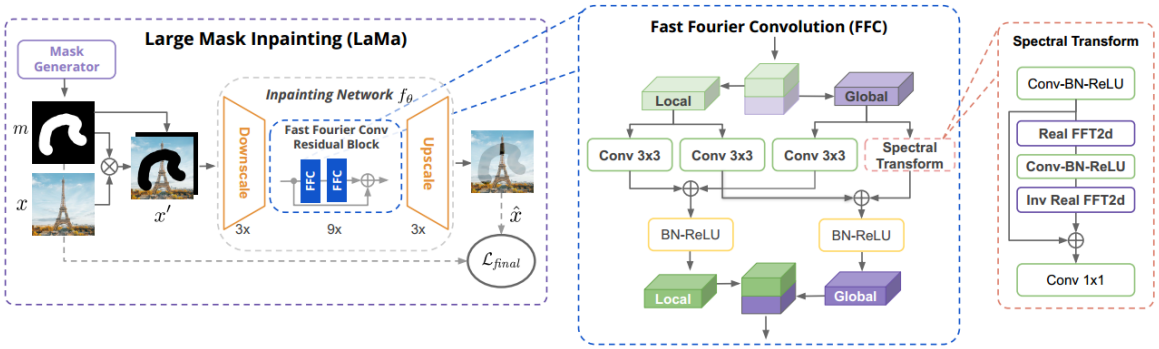


Figure 6: LaMa network architecture from the paper [34]

We download the RGBA undistorted template images for calculating homography and doing substitution from Wikimedia Commons[1]. As the templates had alpha mask, we could create new substitution instance segmentation mask, bounding box and label ground truths for using trained Mask R-CNN model to test the final substitutions. For some sign categories, templates were not available in Wikimedia Commons and for them, we picked the best quality undistorted sample we could find of the category, from the DFG dataset, to use as template.

3.2 Foreground Removal

For foreground removal, we use image inpainting technique which is the task of reconstructing missing regions in an image and it is widely used for image restoration, object removal, compositing etc. Given a RGB image I and its binary mask M of the traffic sign foregrounds, our task is to inpaint the masked region denoted by $I \odot M$.

We tried two methods for this task: OpenCV's implementation of FMM [36] and Large Mask inpainting (LaMa) [34].

LaMa takes as input the stacked masked_image-mask pair, $x = stack(I \odot M, M)$ as a four-channel input and gives the inpainted three-channel RGB image $\hat{x} = f_\theta(x)$ as output, where $f_\theta(\cdot)$ is the LaMa model. The model uses a downsampling block, followed by some residual blocks that use Fast Fourier Convolutions (FFCs) [6] and finally an upsampling block. The network architecture is given in Fig. 6. The advantage of FFC is that it has an image wide receptive field because it uses channel-wise Fast Fourier Transform (FFT). In FFC, channels are split into two parallel branches, one using convolutions (local branch) for local context and the other using real FFT (global branch) for global context. Furthermore, the authors introduced a new loss function involving a high receptive perceptual loss for training.

The foreground removal is important for two reasons, in our pipeline:

1. In our experiments, the original foreground sign sticks out from below the substituted sign if we use the original sign size for placement. This means we will have to increase the size of substitution sign during placement which looked unnatural in many cases. (see Fig. 2)

2. In case we want to replace a traffic sign of one geometry with another, for example, a round sign with a rectangle one, it will be really cumbersome to handle sign placement size.

In our experiments, LaMa provided a realistic inpainting of signs in all the cases while [36] always produced visible artefacts.

[1] https://commons.wikimedia.org/wiki/Category:Road_signs_in_Slovenia

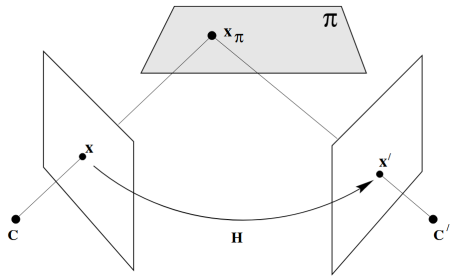


Figure 7: Plane induced homography (Fig. 13.1 in [11])

3.3 Homography estimation

In order to align an undistorted substitution template with the original, we multiply it with the homography matrix $H\_T_k\_k$ calculated between the original sign's template (T_k) and itself (k) (as shown in Fig. 3).

In Fig. 7 we can see the relationship of two image views induced by a world plane. The template and the original sign are two views of the same sign plane π and therefore the points on the template (x) are related to the points on the original sign (x') by a planar homography H induced by the world plane π. The first image plane point x is related to world plane point x_π by the perspectivity $x = H_{1\pi}x_\pi$ while the second image plane point x' is related to world plane point x_π by the perspectivity $x' = H_{2\pi}x_\pi$. The composition of these two perspectivities results in a projectivity and the first image plane point x is related to second image plane point x' by $x' = H_{2\pi}H_{1\pi}^{-1}x = Hx$. We can use H to transform the template view to the original sign view and H^{-1} to do the opposite.

The homography matrix H or as referred to in our work as $H\_T_k\_k$ to imply a transformation between template T_k and original sign k, is a 3×3 matrix with 8 DoF. In order to calculate it, we need a minimum of 4-point correspondences between the two image views of the plane. In this work, we first analyse the efficacy of different keypoint detectors on our traffic sign pairs (see Sec. 4). We experimented with traditional interest point detectors BRISK [17], AKAZE [1] and ORB [27] as well as neural network based SuperPoint [9]. SuperPoint uses a VGG-style [30] shared encoder for dimensionality reduction followed by two heads: Interest Point Decoder for interest point detection and Descriptor Decoder for getting L2-normalized fixed length descriptors. The network architecture as highlighted in the paper is given in Fig. 8. The authors followed a novel self-supervised training regime where they first pre-trained initial interest point detector on synthetic data, applied their proposed Homographic Adaptation procedure to label unannotated data and then trained the full network with both heads on the generated labels.

In order to match keypoints of an image pair, we used a brute-force matcher for BRISK,

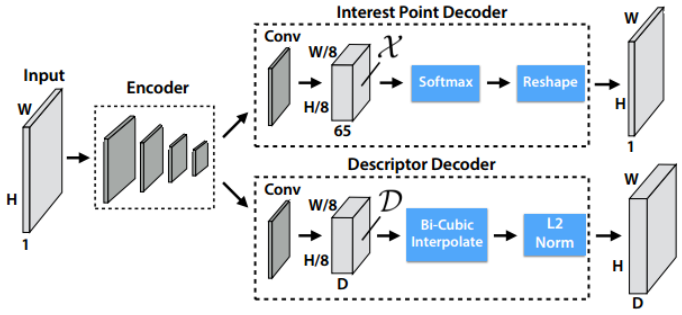


Figure 8: SuperPoint network architecture from the paper [9]

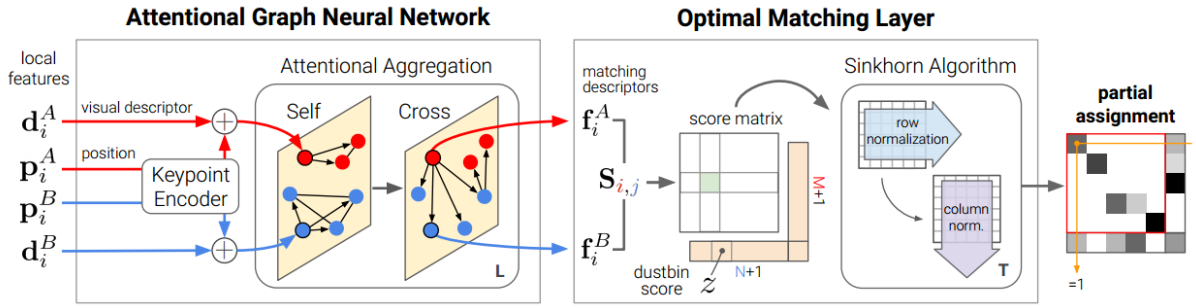


Figure 9: SuperGlue network architecture from the paper [28]

AKAZE and ORB, while for SuperPoint, we used SuperGlue [28]. SuperGlue consists of two main components: Attentional Graph Neural Network and Optimal Matching Layer. Fig. 9 shows the overall SuperGlue architecture as highlighted in the paper. The Attentional Graph Neural Network first uses a Keypoint Encoder to encode keypoint into a high-dimensional vector that later helps the GNN to reason appearance and position jointly. Then a multiplex GNN is used to compute matching descriptors using long-range feature aggregation within (self-edges) and across images (cross-edges). Self-edges are based on self-attention and cross-edges are based on cross-attention. The Optimal Matching Layer first computes score matrix S by using a soft assignment matrix P and maximizing the total score $S * P$. A dustbin is augmented to each set of keypoints to suppress unmatched keypoints resulting in a $(M+1) \times (N+1)$ score matrix with number of features from image A being M and number of features from image B being N. Sinkhorn algorithm normalizes the sum of rows and columns in the score matrix to 1 for T iterations. Finally, the dustbin is dropped and the final partial assignment P is recovered with shape $(M \times N)$.

Once we get the keypoint matches for a sign pair, we calculate homography matrix $H\_T_k\_k$ using DLT. We experiment with both least-squares and RANSAC in the homography computation.

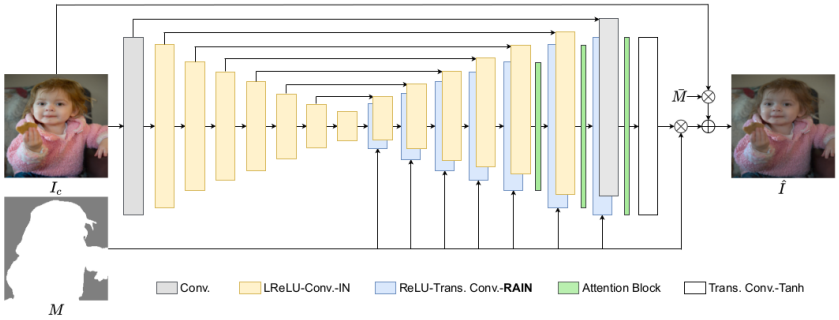
3.4 Composite Blending

Once we transform the substitution template to the correct original sign pose, we use its alpha mask and the instance segmentation mask and bounding box of the original sign to paste the substitution in the correct place. This method results in an abrupt change in the intensity of the foreground and the background and in order to reduce the fuzziness and refine the boundary, we first apply a gaussian blur on the alpha mask to smoothen its edges and then use alpha blending to place the substitution on the inpainted image. The alpha blending is given by the following formula:

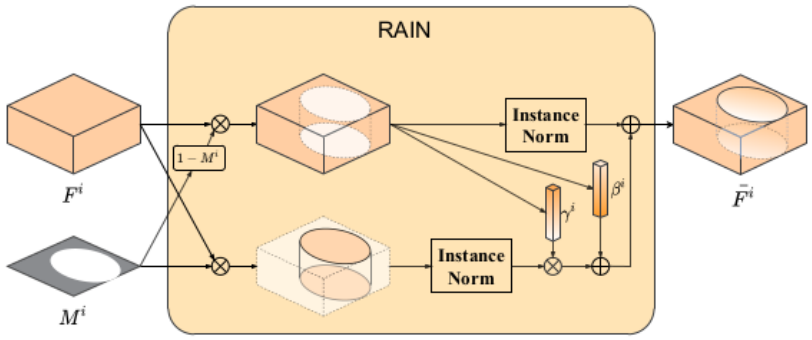
$$output\_img = \alpha * foreground + (1 - \alpha) * background$$

3.5 Composite Harmonization

After all the original foregrounds in the image have been replaced with random substitutions, we need to remove the incompatibility in the appearance between foreground substitutions and the background because the templates are always bright and clear while the backgrounds differ in color and illumination characteristics (eg. season, time of day, weather, etc.). For this, we apply image harmonization which adjusts the composite foreground color and illumination characteristics with that of the background, to make them compatible in appearance.



(a) Network architecture of the generator G



(b) Region-aware Adaptive Instance Normalization (RAIN) module

Figure 10: RainNet [19] network architecture and RAIN module as proposed in their paper

We use RainNet [19] for image harmonization step. It treats the task as a style transfer problem and applies the style information extracted from the background, to the foreground. It consists of a learnable generator G which is a U-Net [26] like model (Fig. 10a) that takes as input the composite image I_c and the foreground mask M and predicts the harmonized image $\hat{I} = G(I_c, M)$ by optimizing $\|G(I_c, M) - I\|_1$ so that the harmonized image remains close to the groundtruth image I. The generator G has Region-aware Adaptive Instance Normalization (RAIN) modules (Fig. 10b) in the decoder which takes as input the convolutional feature F^i and the resized foreground mask M^i of the i-th layer. First, F^i is multiplied with foreground mask M^i and corresponding background mask $1 - M^i$ and then normalized by instance normalization [38]. Following this, the normalized foreground features are affined with learned scale and bias from the background features which results in the new activation value $\bar{F}^i$ at site (h, w, c) in the foreground region. The computation is given by the formula:

$$\bar{F}^i_{h,w,c} = \gamma^i_c \frac{F^i_{h,w,c} - \mu^i_c}{\sigma^i_c} + \beta^i_c,$$

where σ^i_c and μ^i_c are channel-wise variance and mean of the i-th layer foreground feature and γ^i_c and β^i_c are the mean and standard deviation of background activations in channel c of i-th layer. The authors reasoned the efficacy of RAIN normalization for image harmonization over other normalization techniques by pointing that the RAIN module transfers only background statistics to the normalized foreground features without the influences from inconsistent foreground objects.

4 Experiments & Results

For substitution experiments we only consider DFG dataset test split. As reported before in Sec.3.1 and Fig.5, from the set of 200 categories, we exclude the ones that are heterogeneous, since they would have problems with keypoint matching and subsequent homography estimation. We further categorize all signs into super-category of geometry based on their shape: triangles, circles, squares, vertical rectangles, horizontal rectangles and arbitrary (heterogeneous exclusions). Given below is the dictionary of super-categories that we created:

```
categories  = {
            'triangles' : ['I-1', 'I-1.1', 'I-2', 'I-2.1', 'I-3', 'I-4', 'I-5',
                                                    'I-5.1', 'I-5.2', 'I-8',
                    'I-9', 'I-10', 'I-11', 'I-13', 'I-13.1', 'I-14', 'I-15', 'I-
                                                        16', 'I-17', 'I-18',
                    'I-19', 'I-20', 'I-25', 'I-27', 'I-28', 'I-28.1', 'I-29', 'I
                                                        -29.1', 'I-30',
                    'I-32', 'I-34', 'I-36', 'I-37', 'X-4'],
            'circles' : ['II-3', 'II-4', 'II-6', 'II-7', 'II-7.1', 'II-8', 'II-
                                                    10.1', 'II-14', 'II-17',
                    'II-18', 'II-26', 'II-26.1', 'II-28', 'II-30-10', 'II-30-30'
                                                        , 'II-30-40',
                    'II-30-50', 'II-30-60', 'II-30-70', 'II-32', 'II-35', 'II-39
                                                        ',
                    'II-40', 'II-45', 'II-45.1', 'II-45.2', 'II-46', 'II-46.1',
                                                        'II-46.2', 'II-47',
                    'II-47.1', 'III-16', 'III-18-40', 'III-18-50', 'III-18-60',
                                                        'III-18-70',
                    'III-21', 'III-23', 'II-2'],
            'squares' : ['III-1', 'III-2', 'III-5', 'III-6', 'III-8-1', 'III-29-
                                                    30', 'III-29-40', 'III-30-30
                                                    ',
                    'III-35', 'III-107.1-1', 'III-107.1-2', 'III-107-1', 'III-
                                                        107-2',
                    'III-124', 'III-202-5', 'IV-13.1-2',
                    'IV-13-5', 'IV-13-6', 'IV-18', 'I-38', 'II-1', 'III-3'],
            'vertical_rect' : ['I-39-1', 'I-39-2', 'I-39-3', 'III-10', 'III-12',
                    'III-112', 'III-123',
                    'VI-3.1-1', 'VI-3.1-2', 'VI-3-1', 'VI-3-2'],
            'horizontal_rect' : ['III-33', 'III-34', 'III-107.2-1', 'III-107.2-2
                                                    ',
                    'III-120.1', 'IV-12', 'IV-12.1', 'IV-17', 'IV-20-1', 'VI-2.1
                                                        '],
            'arbitrary' : ['II-19-4', 'II-21', 'II-22', 'II-23', 'II-41', 'II-42
                                                    ', 'II-42.1', 'II-43', 'III-
                                                    14',
                    'III-14.1', 'III-15', 'III-25', 'III-25.1', 'III-27', 'III-
                                                        46', 'III-47', 'III-
                                                        50',
                    'III-54', 'III-59', 'III-64', 'III-68', 'III-74', 'III-78',
                                                        'III-84', 'III-84-1'
                                                        ,
                    'III-85.1', 'III-86-1', 'III-86-2', 'III-87', 'III-90', 'III
                                                        -90.1', 'III-90.2',
                    'III-91', 'III-105', 'III-105.1', 'III-105.3', 'III-120-1',
                                                        'III-203-2', 'IV-1',
                    'IV-1.1', 'IV-2', 'IV-3-1', 'IV-3-2', 'IV-3-4', 'IV-3-5', '
                                                        IV-5', 'IV-6', 'IV-
                                                        10',
                    'IV-11', 'VII-4', 'VII-4.1-1', 'VII-4.3', 'VII-4.3-1', 'VII-
                                                        4.3-2', 'VII-4.4-1',
                    'VII-4.4-2', 'VII-4-1', 'VII-4-2', 'X-1.1', 'X-6-3', 'VI-8',
                                                         'II-48', 'II-33', '
                                                        III-37',
```

```
                'III-39', 'III-40', 'III-42', 'III-43', 'III-45', 'III-77',
                                                        'III-85-2', 'III-85-
                                                        3',
                'III-113', 'III-120','III-206-1','IV-13-4', 'IV-13-3','IV-13
                                                        -2','IV-13-1', 'IV-
                                                        13.1-4',
                'IV-13.1-3', 'IV-16','X-1.2', 'II-34']
    }
```

Given below is the number of categories per super-category:

1. **triangles:** 34
2. **circles:** 39
3. **squares:** 22
4. **vertical_rect:** 11
5. **horizontal_rect:** 10
6. **arbitrary:**: 84

Excluding 84 categories in *"arbitrary"* super-category, we experiment with 116 categories from the set of total 200 categories.

We follow the pipeline mentioned in Fig.3. We start with removing existing signs of 116 categories from the testset via inpainting. First we use the instance segmentation annotations of an image to create a binary mask. The binary mask and image pair is given as input to the

Figure 11: Traffic sign removal using big-LaMa [34] model. Left image is original and right image is inpainted result. Notice we only inpaint useful categories

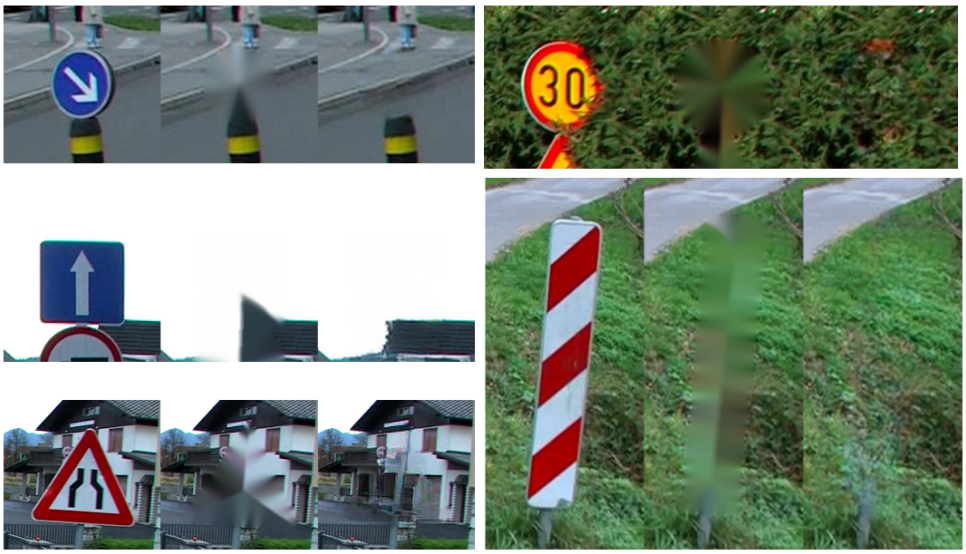


Figure 12: OpenCV's FMM [36] implementation vs Big LaMa [34] model for traffic sign removal via inpainting. For each set, from left: original crop, FMM output, Big LaMa output

Big LaMa network and we get the inpainted result. We use the official Big LaMa model weights trained on subset of 4.5 million images from Places-challenge dataset [43], for inference. We run the model in an offline standalone manner at downsampled 800x800 image resolution (with black padding to maintain original aspect ratio) on Google Colab at batch size 1. Due to GPU constraints we could not test inference on bigger resolutions. The resulting inpainted images are resized to actual 1920x1080 resolution before substitution. Fig.11 shows some inpainted results of Big Lama. We also tried the OpenCV's implementation of Fast Marching Method [36] based inpainting technique but it's always had conical shaped artefacts, as shown in Fig. 12, so we discarded that method and went forward with Big LaMa. Table. 1 shows the Learned perceptual image patch similarity (LPIPS) score of FMM vs Big LaMa (less is better) and we see Big LaMa has slightly better scores.

Method	LPIPS score ↓
FMM [36]	0.00770
Big Lama [34]	**0.00745**

Table 1: LPIPS score of FMM vs Big LaMa for traffic sign removal on DFG testset

In order to understand the efficacy of different keypoint detectors for homography estimation, we first extract all the traffic signs from the whole DFG dataset using segmentation maps, at two image resolutions and create synthetic image pairs to test the average number of detected keypoint matches, number of misses (no keypoint matches) and Mean Average Corner Error (MACE) [8] between target pose and transformed pose. In order to create the synthetic pairs, we use the geometric and photometric augmentation technique used in Theseus library[2] [24] for homography estimation training. First a random geometric transformation is applied followed by random photometric augmentation on the original and the transformed image. The geometric augmentation included rotation, scaling, translation, shearing and perspective transform while the photometric augmentation included contrast, sharpen, exposure, gamma, gaussian smooth, motion blur, shadow highlight, gaussian noise and salt-and-pepper noise. After applying augmentation we have the RGBA original image *img1*, transformed image *img2* and the ground truth homography transformation matrix *H_1_2*. Fig.13 shows few generated pairs of the dataset

[2] https://github.com/facebookresearch/theseus/blob/main/theseus/third_party/easyaug.py

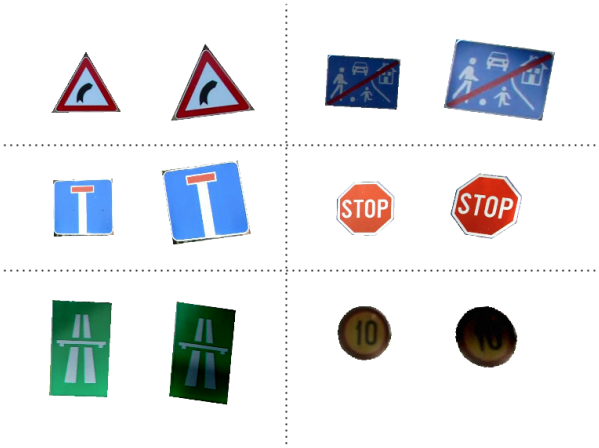


Figure 13: Few examples of augmented pairs created from original DFG traffic sign samples for benchmarking homography estimation methods

we created. Given below are the paramteres of the augmentations:

```
from easyaug import GeoAugParam, RandomGeoAug, RandomPhotoAug
sc = 0.3
rga = RandomGeoAug(
    rotate_param=GeoAugParam(min=-30 * sc, max=30 * sc),
    scale_param=GeoAugParam(min=(1.0 - 0.8 * sc), max=(1.0 + 1.2 * sc)),
    translate_x_param=GeoAugParam(min=-0.2 * sc, max=0.2 * sc),
    translate_y_param=GeoAugParam(min=-0.2 * sc, max=0.2 * sc),
    shear_x_param=GeoAugParam(min=-10 * sc, max=10 * sc),
    shear_y_param=GeoAugParam(min=-10 * sc, max=10 * sc),
    perspective_param=GeoAugParam(min=-0.1 * sc, max=0.1 * sc),
)

rpa = RandomPhotoAug()
prob = 0.2
mag = 0.2
rpa.set_all_probs(prob)
rpa.set_all_mags(mag)
```

First all the extracted traffic signs in the experiments had black background and then we saw having white background produced better matches on the edges and overall better homography estimation. The study of the efficacy of ORB, AKAZE, BRISK and SuperPoint with black background is given in Table.2 and with white background in Table.3. For SuperPoint, we used SuperGlue matcher and least-squares homography estimation while for others, we used brute-force matcher and RANSAC homography estimation. For ORB, we used nfeatures=2000. We used the official weights of SuperPoint trained for outdoor setting and given below are the parameters used for SuperPoint+SuperGlue:

```
--resize : -1 '''Resize the input image before running inference. If two
                 numbers, resize to the exact
                 dimensions, if one number, resize
                 the max dimension, if -1, do not
                 resize'''
--resize_float : True '''Resize the image after casting uint8 to float'''
--superglue : 'outdoor' '''SuperGlue weights'''
--max_keypoints : -1 '''Maximum number of keypoints detected by Superpoint,
                 ('-1' keeps all keypoints)'''
--keypoint_threshold : 0.005 '''SuperPoint keypoint detector confidence
                 threshold'''
--nms_radius : 2 '''SuperPoint Non Maximum Suppression (NMS) radius'''
--sinkhorn_iterations : 20 '''Number of Sinkhorn iterations performed by
```

```
                                                    SuperGlue'''
    --match_threshold : 0.1 '''SuperGlue match threshold'''
```

From Table.[2,3], we see that having white background and cropping from original image size (1920x1080 padded to 1920x1920) for the traffic signs increases homography estimation accuracy greatly and with white background SuperPoint+SuperGlue had zero misses and lowest MACE score. Hence, we decided to go forward with SuperPoint+SuperGlue for homography estimation between original image and it's category template, with white background and we used the default original size 1920x1080 with black padding to 1920x1920 throughout our substitution pipeline.

For an original image and template pair, we first double the size of the original sign since there were many distant signs, and then we resize the template to new resolution of original sign. We set the background of both images to white and then calculate keypoint matches with SuperPoint+SuperGlue using same parameters as mentioned above. In case there are no keypoint matches or if it's a square or rectangle, we take the corresponding four corner points of the rotated bounding boxes of the signs as well. We estimate homography $H\_T_k\_k$ (transformation from template T_k to original sign k) from the matched pairs using least-squares DLT. Least-squares gave better overall results than RANSAC as there were no visible outliers in most cases. Finally we select a substitution sign template randomly of the same super-category (circle for circle, square for square, etc.) and transform it using $H\_T_k\_k$ to have the geometrical characteristics of the original sign k. Fig. 14 shows a few success and failure cases of homography

Resolution	Method	Misses↓	Average Matches↑	MACE↓
800x800	ORB	**766**	**66.903**	**63.1713**
	AKAZE	9453	17.5167	114.1825
	BRISK	2773	28.0403	89.2282
	SuperPoint	887	20.3456	99.1876
1920x1920	ORB	386	**154.1779**	24.7012
	AKAZE	2674	45.9755	35.2064
	BRISK	1174	53.6787	35.8634
	SuperPoint	**47**	50.9648	**10.8202**

Table 2: Efficacy of different keypoint methods for homography estimation on image pair with **BLACK** background. For matching, SuperGlue was used with SuperPoint and BFMatcher was used for all other methods. Best results highlighted in **bold**

Resolution	Method	Misses↓	Average Matches↑	MACE↓
800x800	ORB	**97**	**68.5059**	79.1818
	AKAZE	4080	21.9332	87.3210
	BRISK	895	27.7633	225.5695
	SuperPoint	1008	21.0290	**59.5252**
1920x1920	ORB	32	**162.9177**	12.332
	AKAZE	133	55.1918	23.8899
	BRISK	174	56.1642	44.1530
	SuperPoint	**0**	51.8875	**5.1721**

Table 3: Efficacy of different keypoint methods for homography estimation on image pair with **WHITE** background. For matching, SuperGlue was used with SuperPoint and BFMatcher was used for all other methods. Best results highlighted in **bold**

Figure 14: Homography estimation results of SuperPoint+SuperGlue. From top to bottom: Samples of success cases vs failure cases. From left to right: First image shows the matched keypoints between template T_k and original sign k. Second image shows template, original, transformed template via $H\_T_k\_k$, substitution template, transformed substitution template via $H\_T_k\_k$

estimation using least-squares DLT on SuperPoint+SuperGlue matches.

After we get the transformed substitution template, we first apply a 3×3 Gaussian blur to the alpha channel of the transformed substitution template before using it to alpha blend the foreground to the background. For substitution we use the original foreground's bounding box coordinates for accurate object placement. The Gaussian blur smoothens the edges of the substitution foreground resulting in a more natural substitution.

We now have substitution results and the above steps have taken care of the geometric and semantic inconsistencies mentioned in Fig.2 but the appearance inconsistency is still present because the substitution templates are mostly synthetic or from different image and does not match illumination and color characteristics of the background. In order to rectify the foreground characteristics with the background, we use input the composed substitution result and its alpha mask to RainNet [19] for image harmonization as the last step of the pipeline, using the official weights trained on iHarmony4 [7] dataset. Fig. 15 shows the harmonization and final substitution results of some good samples and Fig. 16 shows some samples where harmonization

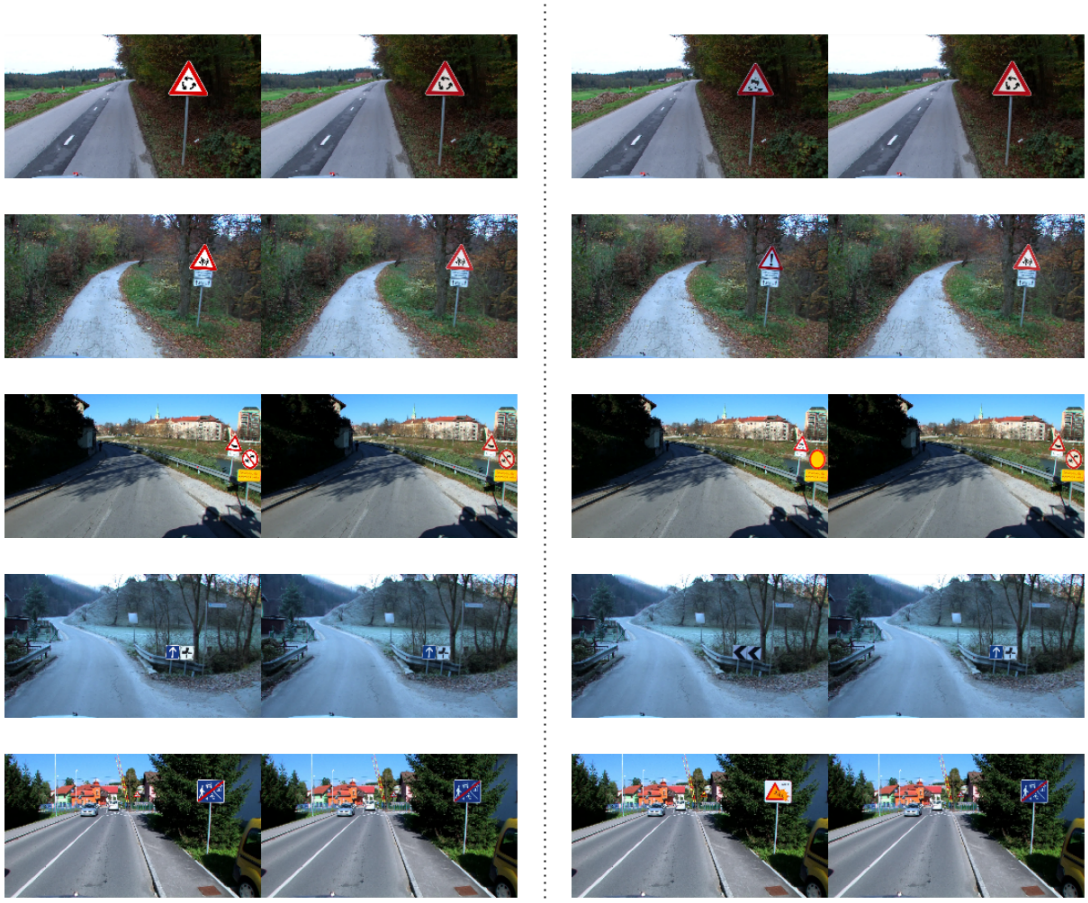


Figure 15: Some good substitution samples. Left: Before and after harmonization. Right: Original image vs final substitution result

failed and final substitution results have big illumination characteristics difference when compared to original and can be visibly identified as fake images. RainNet seems to able to capture the global illumination characteristics of the background but not focus on the local background illumination characteristics surrounding the foreground.

In order to have a quantitative evaluation of the quality of substitution results, we train a Mask R-CNN [12] on the original images and test them on the substitution results for instance segmentation accuracy. We consider only the categories that we've included for substitution (116 out of 200 available categories). For training, we take the official model with ResNet50 [13] backend provided by PyTorch that is pretrained on MS-COCO [18] dataset and modify the pretrained head and mask predictor to support 117 categories (116 + 1 background). Given below are the training parameters:

1. **Image size:** 800 (padded to maintain aspect ratio)
2. **No. of epochs:** 100
3. **No. of classes:** 117
4. **Batch size:** 8
5. **Optimizer:** SGD (lr=0.005, momentum=0.9, weight_decay=0.0001)
6. **LR Scheduler:**: StepLR (step_size=30, gamma=0.05)

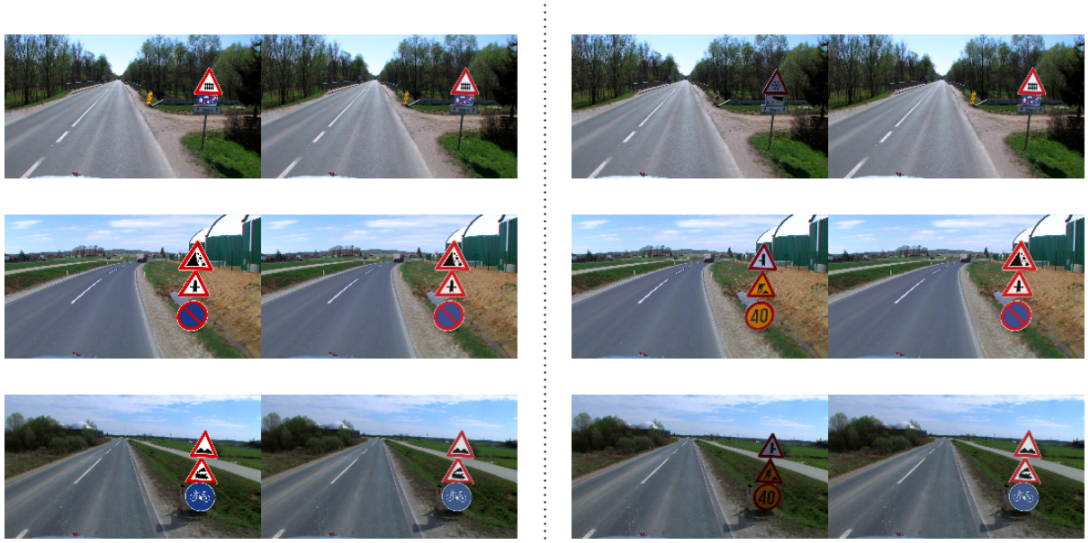


Figure 16: Some bad substitution samples due to ineffective harmonization. Left: Before and after harmonization. Right: Original image vs final substitution result

7. **Augmentations:**: RGBShift, RandomBrightnessContrast, MotionBlur, MedianBlur, GaussianBlur, Sharpen, Emboss, HueSaturationValue

Since there was no seperate validation set on DFG Traffic Sign Dataset [35], we made a 80:20 split on the provided training split to make a new train:val split. After removal of heterogeneous categories, we had 3068 training, 767 validation and 1260 test images. We picked the model weight that had the best validation accuracy. The epoch 35 weights had the best validation accuracy and Table.4 shows the bounding box and segmentation mAP (mean Average Precision) of said weight on original test images and on same test images with substitution.

Split	mAP@0.5	mAP@0.75	mAP@0.50:0.95
Test org bbox	0.836	0.819	0.732
Test subst bbox	0.709	0.684	0.556
Test org segm	0.834	0.810	0.743
Test subst segm	0.709	0.701	0.636

Table 4: mAP of bounding box and segmentation under different IoU settings on original test images and their corresponding substitution results

Even though the substitutions look natural, there is a noticeable drop in mAP on substitution images. We think this drop is because of homography and harmonization failures and low generalization capability of our trained model under synthetic substitution settings as we did not use any synthetic copy-paste or similar augmentations while training.

Except image inpainting, which was done on Google Colab, all other experiments were done on Ryzen 7 CPU and GTX 1070 hardware.

5 Conclusion

In this paper, we've introduced a Plug-and-Play pipeline with modular components for traffic sign substitution and we obtained natural looking results in most cases. We've divided the issues faced in image composition into different components in our pipeline and each component focuses on a separate research domain. We can replace any component later with other SOTA

models, without affecting other components. Our experiments revealed errors in homography as the main cause of unnatural looking substitution, followed by errors in image harmonization. A closer inspection revealed small resolution of foreground and lack of details in texture as the reason behind errors in homography. This work used official trained models of the papers we used and we feel retraining them on a traffic sign dataset will result in much better results, especially on low-resolution traffic signs with low details in their textures. In future, we will extend this work on finding effective ways and networks to train on traffic sign datasets and vividly exploring the cause of failures, including the dip in accuracy of the instance segmentation model.

References

[1] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298, 2011. 3, 8

[2] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI 12*, pages 214–227. Springer, 2012. 3

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 3

[4] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001. 3

[5] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*, pages 778–792. Springer, 2010. 3

[6] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. *Advances in Neural Information Processing Systems*, 33:4479–4488, 2020. 3, 7

[7] Wenyan Cong, Jianfu Zhang, Li Niu, Liu Liu, Zhixin Ling, Weiyuan Li, and Liqing Zhang. Dovenet: Deep image harmonization via domain verification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8394–8403, 2020. 16

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016. 4, 13

[9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 1, 3, 8

[10] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 682–691, 2019. 4

[11] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. 8

[12] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 17

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 17

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 4

[15] Daniela Horn and Sebastian Houben. Fully automated traffic sign substitution in real-world images for large-scale data augmentation. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 465–471, 2020. 2, 4, 5

[16] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726*, 2017. 3

[17] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee, 2011. 8

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 1, 17

[19] Jun Ling, Han Xue, Li Song, Rong Xie, and Xiao Gu. Region-aware adaptive instance normalization for image harmonization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9361–9370, 2021. 1, 4, 10, 16

[20] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 3

[21] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. 2019. 3

[22] Li Niu, Wenyan Cong, Liu Liu, Yan Hong, Bo Zhang, Jing Liang, and Liqing Zhang. Making images real again: A comprehensive survey on deep image composition. *arXiv preprint arXiv:2106.14490*, 2021. 3, 5

[23] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, pages 313–318. 2003. 4

[24] Luis Pineda, Taosha Fan, Maurizio Monge, Shobha Venkataraman, Paloma Sodhi, Ricky TQ Chen, Joseph Ortiz, Daniel DeTone, Austin Wang, Stuart Anderson, Jing Dong, Brandon Amos, and Mustafa Mukadam. Theseus: A Library for Differentiable Nonlinear Optimization. *Advances in Neural Information Processing Systems*, 2022. 13

[25] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, 1984. 4

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 10

[27] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. 2, 3, 8

[28] 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. 1, 4, 9

[29] Tiago A Schieber, Laura Carpi, Albert Díaz-Guilera, Panos M Pardalos, Cristina Masoller, and Martín G Ravetti. Quantification of network structural dissimilarities. *Nature communications*, 8(1):13928, 2017. 4

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 8

[31] Konstantin Sofiiuk, Polina Popenova, and Anton Konushin. Foreground-aware semantic representations for image harmonization. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1619–1628, 2021. 4

[32] Dominic Spata, Daniela Horn, and Sebastian Houben. Generation of natural traffic sign images using domain translation with cycle-consistent generative adversarial networks. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 702–708, 2019. 2, 4, 5

[33] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0), 2012. 2

[34] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022. 1, 3, 7, 12, 13

[35] Domen Tabernik and Danijel Skočaj. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. *IEEE Transactions on Intelligent Transportation Systems*, 2019. 1, 6, 18

[36] Alexandru Telea. An image inpainting technique based on the fast marching method. *Journal of graphics tools*, 9(1):23–34, 2004. 3, 7, 13

[37] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, Ambrish Tyagi, James M Rehg, and Visesh Chari. Learning to generate synthetic data via compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 461–470, 2019. 4

[38] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 10

[39] Qing Wang, Jiaming Zhang, Kailun Yang, Kunyu Peng, and Rainer Stiefelhagen. Matchformer: Interleaving attention in transformers for feature matching. In *Proceedings of the Asian Conference on Computer Vision*, pages 2746–2762, 2022. 4

[40] Huikai Wu, Shuai Zheng, Junge Zhang, and Kaiqi Huang. Gp-gan: Towards realistic high-resolution image blending. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2487–2495, 2019. 4

[41] Lingzhi Zhang, Tarmily Wen, and Jianbo Shi. Deep image blending. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 231–240, 2020. 4

[42] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 4

[43] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017. 13

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2