Semantically-Aware Image Extrapolation

Anonymous CVPR 2021 submission

Paper ID 9939

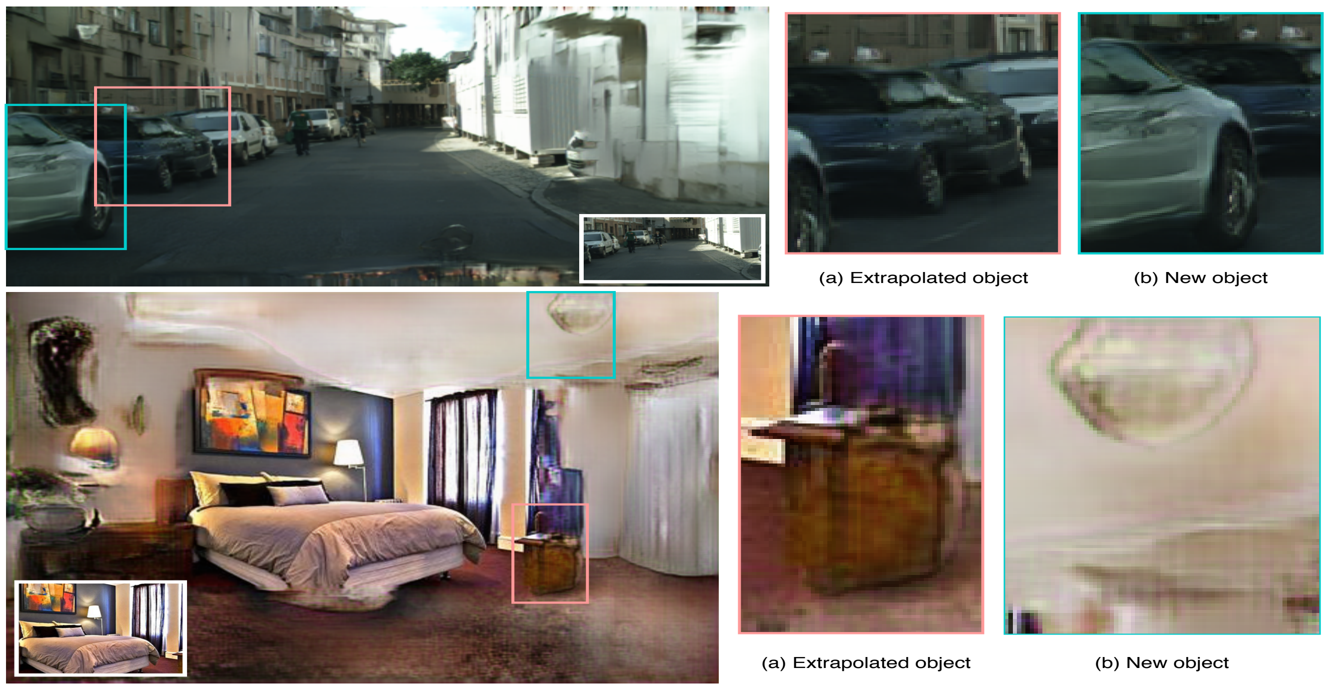

Figure 1: We propose a novel technique for image extrapolation by editing in the semantic label space. Our method not only extrapolates the objects present in the input but also generates new objects. As shown above, the input image is extrapolated on all the 4 sides to double both the dimensions. Our method not only adds new objects to the outpainted region, but also maintains the texture consistency giving a realistic outlook to the generated image.

Abstract

*We propose a semantically-aware novel paradigm to perform image extrapolation that enables the addition of new object instances. All previous methods are limited in their capability of extrapolation to merely extending the already existing objects in the image. However, our proposed approach focuses not only on **(i)** extending the already present objects but also on **(ii)** adding new objects in the extended region based on the context. To this end, for a given image, we first obtain an object segmentation map using a state-of-the-art semantic segmentation method. The, thus, obtained segmentation map is fed into a network to compute the extrapolated semantic segmentation and the corresponding panoptic segmentation maps. The input image and the obtained segmentation maps are further utilized to generate the final extrapolated image. We conduct experiments on Cityscapes and ADE20K bedroom datasets and show that our method outperforms all baselines in terms of FID, and similarity object co-occurrence statistics.*

1. Introduction

Image extrapolation or out-painting refers to the problem of extending an input image beyond its boundaries. While the problem has applications in virtual reality, vertically filmed video expansion, sharing photos on social media like Instagram, and even generating scenes during game development especially if the scenes are repetitive, it is relatively under-explored compared to the image inpainting counterpart which has been extensively researched in recent years. Image in-painting solutions based on deep networks and generative adversarial networks (GANs), when applied to the out-painting problem, have been shown to yield poor results [30]. This has led to researchers exploring and proposing new solutions to the out-painting problem [41, 35, 32]. However, the solutions have been mainly restricted to images that involve outdoor domains like natural scenes where the problem is limited to just extending the existing textures for 'stuff' classes like mountains, water, trees. These methods are not suitable to other domains like traffic scenes and indoor scenes where the image extrapolation involves 1) extending not only the 'stuff' classes but also the 'things' classes like cars, persons, beds, tables that have very definite structure and 2) adding new objects that were not present in the original image. The current techniques, however, fail to address the above two requirements and are, therefore, limited in their ability to perform satisfactory image extrapolation.

To this end, we address the shortcomings of the previous works in image out-painting by extrapolating the image in the semantic label map space. We first use a state-of-the-art object segmentation algorithm to obtain dense pixel-wise object label maps for the input image. We, then, extrapolate the thus obtained label map using a generative adversarial network [10] framework. Using label map enables us to extend the existing objects in semantically aware fashion to the extent that is required as well as adding new objects based on the context. However, label maps can only encode pixel-wise class information and cannot distinguish between different instances of the same class. In order to create multiple instances of the same class, we generate panoptic label map from the segmentation map. This provides us with the capability to have control over each instance separately. The input image and the extrapolated label maps (class segmentation and panoptic) are fed to another GAN based network to obtain the final out-painted image. We believe our method, due to its ability to leverage semantic label space for extrapolation, is a significant departure from the general methods that are used for image extrapolation which mainly operate in pixel domain.

Our contributions can be summarized below:

- We propose a novel paradigm for image out-painting by extrapolating the image in the semantic label space.
- We show how one can generate novel objects by generating the extrapolated semantic label map.
- We propose the generation of panoptic label maps from the extrapolated semantic label maps to facilitate the generation of multiple instances of the same class.
- Through extensive experiments, we show that our method outperforms all previous state-of-the-art methods in image out-painting in terms of FID and similarity in object co-occurrence metrics. We further show in our method, the effect of using the generate panoptic label maps to generate crisp object boundaries in the final extrapolated image.

2. Related Work

Prior works in image synthesis have had great breakthroughs in Image inpainting [21, 36, 38] , conditional image synthesis [12, 22, 26, 31, 33], and unconditional image synthesis [1, 24, 7]. On the contrary, image extrapolation models have been relatively less successful. The works on image extrapolation can be broadly classified on whether they use non-parametric methods or are learning based i.e. parametric methods. Several non-parametric methods [8, 9] have been able to perform only a limited peripheral texture extension. Furthermore, their heuristics don't capture the variation in color, texture and the information of shape and structure of an object. These methods [9] limit themselves to simple pattern extrapolation and are very brittle to increasing extrapolation. Other classical approaches [3] use patch matching and try to extrapolate on the basis of the input image. Considering the fact that the extrapolated section of the image in a general case can be different to the input image section in terms of texture and objects, these methods have an inherent limit of information. Moreover, since they fetch information from the input images and are unable to fit in the non-linearity with the simple heuristics, these works tend to replicate the internal patches for complex sections creating artifacts in the extrapolated regions.

With the advent of GAN [10] based approaches, significant progress has been made in image extrapolation. [41, 35, 30, 32] use a single stage method to extrapolate the input image. Most of these works deal with scene completion using object completion or merely extending the significant texture near the image boundary. Consequently, they quickly breakdown upon further extrapolation of the input. As we increase the section to be extrapolated, the relative volume information in input reduces. Relatively reduced prior information in input, results in generation of substandard extrapolation. Moreover, all of these approaches currently lack the semantic understanding of the scene. This limits their ability to merely repeating a continuous scene in the extrapolated region, without generating concomitant novel objects.

Conditional Image Synthesis: In our work, we make use of the advances in conditional image synthesis for the last stage of our algorithm as discussed later. Pix2Pix [12], pix2pixHD [31] and SPADE [26] use conditional GANs to synthesize images from their semantic label maps. We also draw inspirations from conditional image synthesis to our semantic label map extrapolation stage conditioned on the input semantic label map.

Object generation: Only a few works have focused on the creation of novel objects in the edited image. Azadi et al. [2] concentrate on synthesizing images using semantic label space but in an unconditional setting, where the semantic label space is generated from scratch from a random seed. For a given sparse scene, [17] focus on adding objects of similar class in the scene to densely populate the scene.

3. Our Method

Image out-painting methods based on deep networks have leveraged image-to-image translation approach. This approach leads to a mere extension of the peripheral textures in the image. We argue that direct image-to-image translation is the reason for their limited success. Instead, we take a different approach to the past works and incorporate the usage of semantic label map for extrapolation. Label maps lie on a much lower manifold as compared to natural images. This gives a hint that it might be easier to tackle the image extrapolation problem in the semantic label space and quickly learn object semantics, class information, and definitive shapes of objects as opposed to the previous works. Working in this solution space enables us to extract essential features in form of segmentation maps and panoptic maps.

Our goal is to extrapolate a given image $X(\in \mathbb{R}^{h\times w\times c})$ on its periphery using a deep generative neural network based pipeline. The generated image is $Y(\in \mathbb{R}^{h_1\times w_1\times c})$ where $h_1 \geq h$ and $w_1 \geq w$. Here, c represents the number of channels corresponding to the image, which is 3 for a colored image. The pipeline involves four major stages:

- Image segmentation: Generation of the segmentation map corresponding to the input image.
- Semantic label map extrapolation: Generation of the peripheral extended region in the semantic label map space.
- Panoptic label generation: The semantic label map is further segmented to create multiple instances of the same object class.
- Instance-aware image synthesis: Conversion of the extrapolated semantic label map to the corresponding colored image.

3.1. Image Segmentation

Given an image $X(\in \mathbb{R}^{h\times w\times c})$, corresponding one-hot vector for semantic label map $L(\in \{0,1\}^{h\times w\times c_1})$ is obtained using state-of-the-art segmentation techniques [42, 29, 4, 40, 37]. The resultant semantic label map generated contains c_1 channels where each channel corresponds to one class. The semantic label map, thus generated, is extrapolated in the next stage.

3.2. Semantic Label Extrapolation

We use the semantic label map obtained from the previous stage to synthesise a new extrapolated label map. The label map L, obtained from the previous stage is zero-padded to form a new label map $L_1(\in \{0,1\}^{h_1\times w_1\times c_1})$. This label map L_1 is extrapolated to the final label map $L_2(\in \{0,1\}^{h_1\times w_1\times c_2})$, where $c_2 = c_1 + 1$. The obtained label map contains an extra channel that corresponds to the boundary map of the extrapolated label map.

Generator

We used SPADE [26] residual blocks for each of the layers in the generator. These residual blocks tend to retain more information from the input image and thereby empower the network. Further the input image is fed to each of the layer along with the computed values of the previous layer in the network. We use spectral normalization [25] for each of the generator layer.

With the first c_1 channels, the model had the semantic label map information only. Segmentation maps do not distinguish between two objects of the same class, occurring adjacent to each other. This poses a difficulty for the network while inferring shape information from the segmentation map. While training, the ground truth label map of the image is concatenated with an extra channel corresponding to the object instance boundary map. Adding the extra boundary map channel ensures that the generated segmentation map corresponds to the ground truth boundary and incorporate better object shape information.

At this stage, we use LS-GAN loss [23] ($\mathcal{L}_{GAN}$), Feature matching loss [31] based on the discriminator ($\mathcal{L}_{FM}$). We use focal loss [20] ($\mathcal{L}_{FL}$) between the generated label map and ground truth label map to penalize discrepancy between them as shown below. We use an additional cross entropy loss ($\mathcal{L}_{CE}$) between the ground truth instance boundary map and the boundary map channel of the generated semantic image, as shown below.

$$l(z,y) = -y \times log(z)$$

$$\mathcal{L}_{CE}(z,y) = \Sigma_{h,w,c} l(z,y)$$

$$\mathcal{L}_{FL}(z,y) = \Sigma_{h,w,c} l(z,y) \times (1-z)^{\gamma}$$

We use the following training objective for semantic label map extrapolation:

$$\min_G (\mathcal{L}_{GAN} + \mathcal{L}_{FM} + \lambda_{FL}\mathcal{L}_{FL} + \lambda_{CE}\mathcal{L}_{CE}) \quad (1)$$

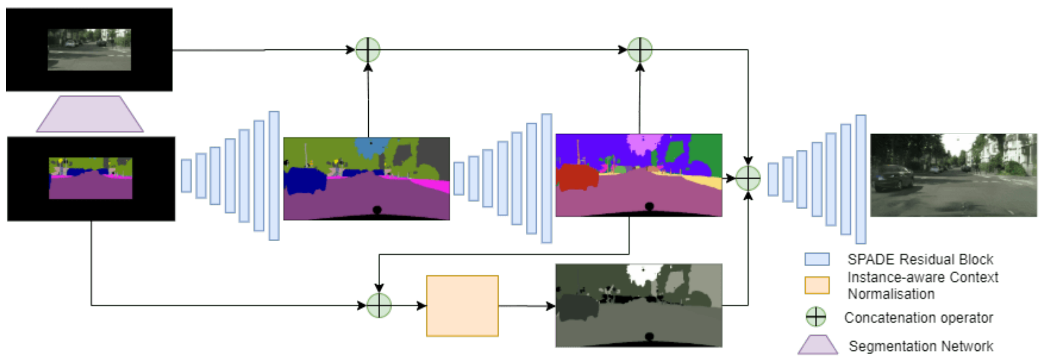

Figure 2: Overview of our pipeline: The input image is fed into a pre-trained segmentation network to obtain a label map. The label map is fed into our semantic extrapolation generator to obtain a label map that is four times the input image. The extrapolated label map is input into another network to obtain panoptic label map to create instance map. The extrapolated label map, the panoptic label are used in conjunction with our instance-aware context normalization to obtain the final image extrapolated image.

Discriminator
Our method involves generation of high resolution image. Thus, a single scale discriminator fails to capture both the low frequency and high frequency details in the image to differentiate between synthesized and real image. Thus, we use a multi-scale discriminator, as used in pix2pixHD [31], which efficiently discriminates between real and fake generations in high-resolution images. The multi-scale discriminator works at three different scales of the image. Each of the scale differs by a factor of 2. While the discriminator at the finest scale operates on the high frequency details in the image, the low frequency details are operated on by the highest scale discriminator. This helps the model generate images at a higher resolution.

3.3. Panoptic Label Map Generation

One of the key ingredients of a good image extrapolator is to generate crisp and precise boundaries between object instances of the same class. To this end, we generate object instance maps from the extrapolated segmentation maps. [4] proposed to generate panoptic label maps from natural images. However, we do not have the extrapolated image to be able to generate the panoptic labels. We circumvent this by adapting the method elucidated in [4] by predicting the class-agnostic instance centers and pixel-wise offsets from the centers of the instances they belong to. Specifically, we train a generator-only network that takes in the extrapolated segmentation map and produces heat maps for instance centers and the pixel-wise offsets from the nearest instance center. The center heat-maps and the offset outputs are further processed along with the segmentation map to obtain the instance maps. The details of training of the network and post-processing are provided in the supplementary material.

3.4. Instance Aware Image Synthesis

This is the final stage which converts the extrapolated semantic label map back into a colored image. This stage takes in input $X'(\in \mathbb{R}^{h_1 \times w_1 \times c'})$ (Figure 2), which is a concatenation of the extrapolated semantic label map obtained from the second stage, the cropped (input) image, the boundary map obtained using the panoptic label map obtained from the previous stage and the feature map obtained using Instance-aware Context Normalization, discussed later. Hence, $c' = c_1 + 3 + 1 + 3 = c_1 + 7$, where c_1 is the number of classes in the corresponding semantic label map. The output is an RGB image $Y \in \mathbb{R}^{h_1 \times w_1 \times 3}$

This is different from prior conditional GANs problems [12, 22, 26, 31] since they synthesize RGB images from semantic label maps, but we have to synthesize RGB images from semantic label maps, given some pixel information of the to-be-synthesized RGB image, in this case it is the cropped image. Here, we have to take care of texture consistency of the synthesized image while maintaining an identity mapping from the cropped image to the final image. To maintain this texture consistency, we concatenate the feature maps to the input, which are generated using Instance-aware Context Normalisation module, discussed in the subsequent sections.

Generator
We used SPADE [26] normalization residual blocks for each of the layers in the generator. We use similar learning objective functions, as used in SPADE [26] and pix2pixHD [31]; GAN loss with hinge-term [19, 25, 39] ($\mathcal{L}_{GAN}$), Feature matching loss [31] based on the discriminator ($\mathcal{L}_{FM}$) and VGGNet [28] for perceptual losses [6, 13] ($\mathcal{L}_{VGG}$)

Instance-aware Context Normalization
Outpainting-SRN [32] proposed Context Normalization (CN) to maintain texture consistency between the inside (cropped) region and the outside (outpainted) region. It in-

volves transferring the mean feature or color from the inside region to the outside region. However, we believe that transferring this input mean color directly to the outside region is not for suitable images which have very diverse object instances (like outdoor images, street images).

To this end, we propose Instance-aware Context Normalization (IaCN) module (Figure 2), which takes as input the input cropped image and the instance map. IaCN module computes the mean color using the input (cropped) image for all the partial instances. Partial instances refer to the instances which get extrapolated in the final image. Since the problem with texture consistency occurs only for partial instances, therefore we compute features only for partial instances. These computed feature maps are then concatenated to the input.

Image Discriminators
We propose to use two discriminators, i) a traditional image discriminator (multi-scale discriminator) that attempts to differentiate between the real and the fake image, ii) a patch co-occurrence discriminator similar to swapping autoencoder [27]. Park et al. employed a co-occurrence patch discriminator to ensure texture transfer [14, 34] from an input image to the target image to be edited. We employ a similar idea wherein the region in the image that needs to be extrapolated takes the role of the target image (equation 2). This facilitates the consistent texture transfer consistently from the inside region to the extrapolated region (illustrated in Figure 3).

$$\begin{aligned}&\mathcal{L}_{CooccurGAN}(G, D_{patch}) = \\ &\mathbb{E}_{x,y}[-log(D_{patch}(crop(G(x)), crop(y), crops(y)))]\end{aligned} \tag{2}$$

Here x is the input and y is the corresponding ground-truth image. $crop(y)$ function takes random patches of 64×64 from image y and $crops(y)$ takes 4 random patches from image y, which serve as the reference patches. The details of the network architectures for all generators and discriminators for the various stages are provided in the supplementary material.

Variational Autoencoder
To ensure appropriate style transfer, we use an encoder that processes the cropped image, which is then fed to the generator. We use the encoder used in SPADE [26]. This encoder forms a VAE [16] with the generator. In the objective function, we add a KL-Divergence Loss term [16] ($\mathcal{L}_{KLD}$).

Final Objective
The training objective is as shown below in equation 3:

$$\begin{aligned}\min_G(\mathcal{L}_{GAN} + \lambda_{FM}\mathcal{L}_{FM} + \lambda_{VGG}\mathcal{L}_{VGG} + \\ \lambda_{KLD}\mathcal{L}_{KLD} + \mathcal{L}_{CooccurGAN})\end{aligned} \tag{3}$$

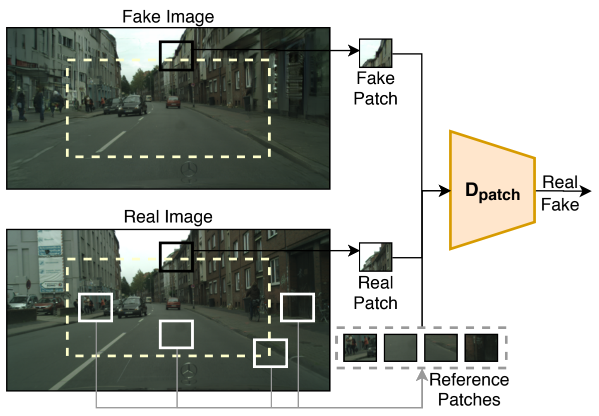

Figure 3: Patch Discriminator: D_{patch} takes in input 4 reference patches, a fake patch and a real patch. The reference patches are randomly selected from the real image. The fake patch and real patch are the same patches, randomly selected but made sure that some part of them is inside while other part is outside, from fake image and real image respectively. The discriminator tries to distinguish between fake patch and the real patch, making use of the reference patches. All the patches are of size 64×64.

4. Experiments

We evaluate the proposed approach on two different datasets which have a sufficient disparity between each other to show that our approach is fairly robust and is applicable to diverse scenes. We utilize the publicly available Cityscapes [5] and ADE20K [43] bedroom subset both of which were originally proposed for semantic segmenation. While Cityscapes comprises of outdoor street images, ADE20K bedroom subset consists of bedroom scenes. To obtain the processed subset, we contacted the authors of [17]. The processed subset consists of 31 classes including bed, lamp, wall, floor and table. Cityscapes consists of 2975 training images and 500 validation images. Each image has its corresponding semantic label map and instance label map along with the original image. The bedroom subset of ADE20K [43] has 1389 images in the training set and 139 in the validation set. In order to limit the size of our model, we downsample the images in Cityscapes to a resolution of 256×512 and the ADE20K bedroom by resizing all its images to a standard size of 384×512 while training.

Implementation details As discussed, we used the standard semantic segmentation models for our first stage which involves generation of semantic map of the input image. We train PSPNet [42] on Cityscapes as well as ADE20K bedroom subset at the resolution discussed earlier and use them to generate segmentation maps of the input (cropped) images.

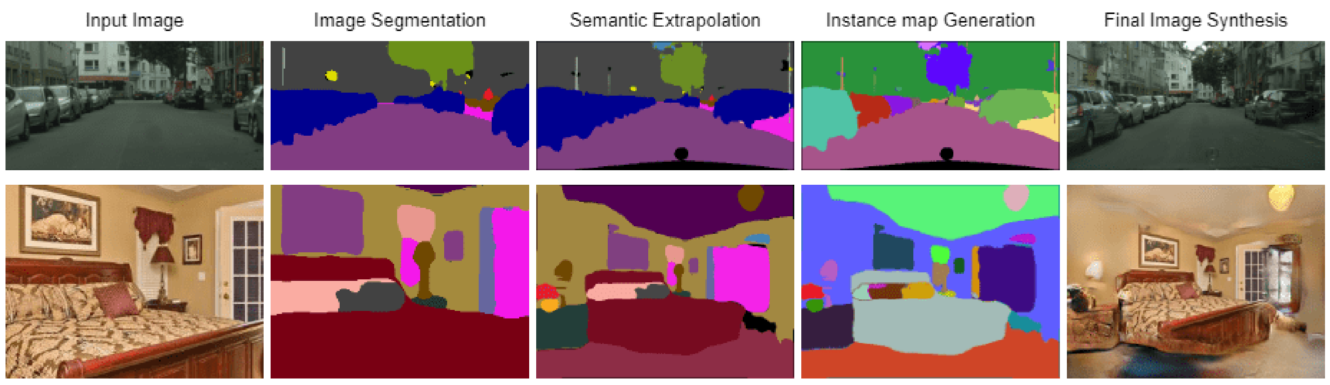

Figure 4: Stage-wise results for Cityscapes and ADE20K dataset. The input (cropped) image is converted to semantic label map in stage-1, which is then extrapolated in stage-2 to form a new (outpainted) semantic label map. Instance maps are generated from this semantic label map in stage-3. The input image, the (outpainted) semantic label map and the instance map are used to synthesize the final image in stage-4.

For the training of stage-2, in our final objective (Equation 1), we use $\lambda_{FL} = 5$, $\lambda_{CE} = 5$ and $\gamma = 5$. For the training of stage-4, we use the same weighting for loss terms as SPADE [26], i.e. $\lambda_{FM} = 10, \lambda_{VGG} = 10$ and $\lambda_{KLD} = 0.05$ in Eq 3. We use ADAM solver [15] with $\beta_1 = 0$ and $\beta_2 = 0.9$ for both the stages. The training is done for 200 epochs.

Baselines We compare our method with various baselines both in quantitative (with FID and Similarity in Object Co-Occurrence metrics) and qualitative terms. We compare the proposed approach with three baselines 'Outpainting-SRN' [32], 'Boundless' [30] and partial convolutions ('PConv') [21]. For [32], we adapt the official public code repo made available by the authors. Although, 'PConv' [21] was originally proposed for image in-painting, like in [30], we adapt it for the image out-painting. We also note that none of these baselines are trained on our kind of input-crop (25% of the original image), thus we train all of them using similar inputs and outputs as in our method. For each of the baseline, we evaluate their ability to generate the complete image from the central part consisting of 25% area of the actual size of the image. The hyper-parameter details for each one of these baselines is provided in the supplementary material.

Evaluation Metrics We evaluate our model and compare against baselines on two metrics - Frêchet Inception distance (FID) [11], and similarity in object co-occurrence (SOCC) statistics [18].

FID: It is a standard metric used to calculate the fidelity of the GAN generated images and provides a measure of the distance between the generated images and the real images in the dataset. Such distance is calculated in the feature space of the Inception network output. The pretrained Inception model is used to produce the features corresponding to each image - real as well as generated. The distance is then calculated using the Frêchet distance formula $||m_r - m_g||^2 + Tr(C_r + C_g - 2\sqrt{C_r C_g})$

SOCC: Since our method is aimed at incorporating the object co-occurrences to generate new objects while extrapolating, we test this ability with the help of similarity co-occurrence measure as proposed in [18]. The co-occurence measure for two classes c_a and c_b can be calculated as the ratio of the number of times they occur together to the total number of times one of them occurs in the entire dataset. Let N_{c_a} represent the frequency of a class c_a in the input image, and $N_{c_{ab}}$ be the number of times there is atleast one instance of class c_b present in the extrapolated region given c_a is present in the input. The probability of co-occurrence $p(c_a, c_b)$ of the two classes can be calculated as $\frac{N_{c_{ab}}}{N_{c_a}}$. The similarity in co-occurrence probability of a pair of classes between generated outputs and the training set, therefore, reflects the extent of faithful emulation of scene distribution. The similarity in co-occurrence for class c_2 in the output to the training set given c_1 is present in the output is

Method	Cityscapes	ADE20k
Pconv	86.82	147.14
Boundless	77.36	136.98
Outpainting-SRN	66.89	140.98
Ours	**47.67**	**90.45**

Table 1: Results: FID scores (lower is better) for our method vs the baselines on Cityscapes and ADE20K-bedroom dataset. Our method outperforms other baselines by a significant margin in both the datasets.

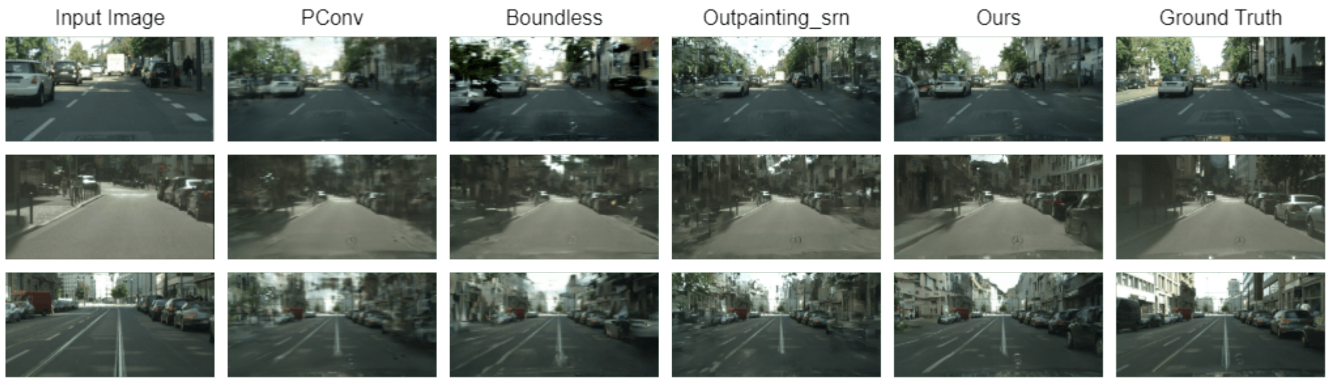

Figure 5: Visual comparison between our model and baselines on the cityscapes dataset. Our method is able to add objects in the outpainted image, a black car (left part of the image) in the first row, and a black car (right part of the image) in the last row. While Boundless [30] tries to create new objects, our method creates far detailed and accurate object Pconv [21] and Outpainting_srn [32] don't produce new objects in the extrapolated image.

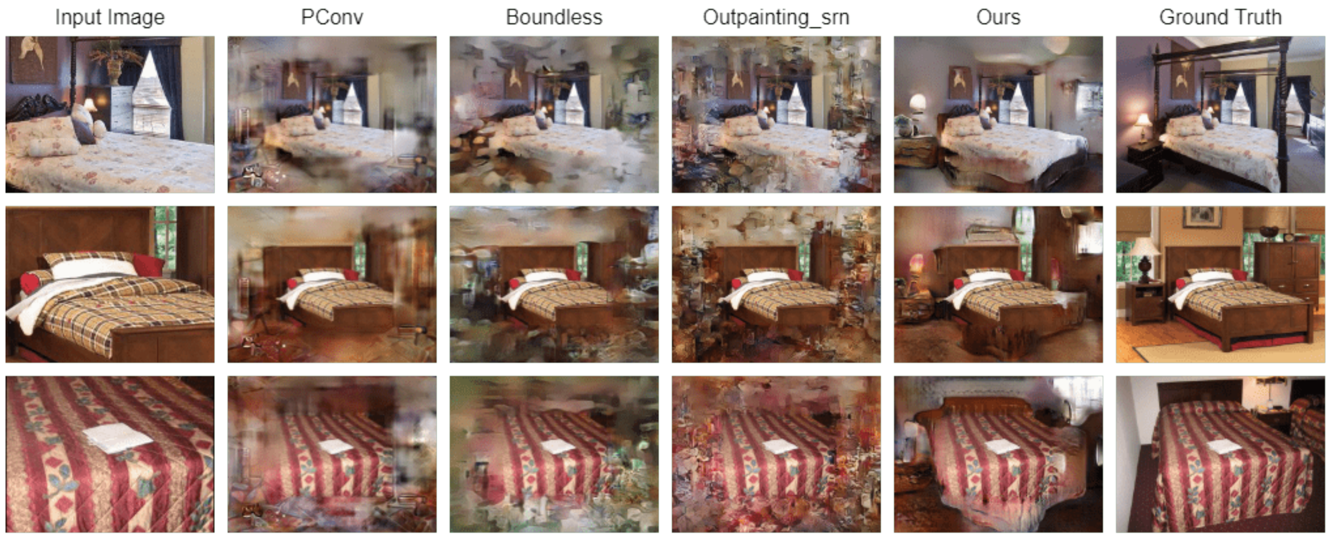

Figure 6: Visual comparison between our model and baselines on the ADE20K-bedroom dataset. Our method generates new objects (bedside lamps) in all 3 images and produces more realistic results compared to baseline methods. Pconv [21] tries to smoothen the input image whereas, Boundless [30] and Outpainting_srn [32] generate lots of artifacts in the outpainted image.

defined as $s(c_a, c_a) = 1 - |p_{train}(c_a, c_b) - p_{gen}(c_a, c_b)|$. The closer is this score to 1, the greater is the similarity between the outputs of the model and the training set images.

4.1. Qualitative performance

In figure 4, we show the various stages of our pipeline. The dense label map obtained by feeding the image into PSPNet is shown in column 2. The thus obtained label map is fed into the proposed semantic label extrapolation network as shown in column 3. In row 1, the presence of a sequence of cars in the input label maps results in more cars being added and thus guarantees the continuity in the label space. Similarly in row 2, our semantic label extrapolation network exploits the presence of the bed in the input and generates a side-table and a table lamp, thus ensuring semantic continuity. We further obtain instance maps from the extrapolated label maps using our instance map generator. This enables to generates definitive object boundaries as well facilitates rich and diverse texture composition when there are more instances of the same class. The effect of this is pronounced in row 1, wherein multiple instances of cars are generated from big blobs in the extrapolated label maps.

In figure 5, we compare our results with the baselines for

Method	(Parking, Car)	(Person, Person)	(Pole, Traffic Light)	(Person, Rider)	(Car,Sidewalk)	Average
Outpainting-SRN	0.85	0.89	0.68	0.91	0.85	0.94
Boundless	0.82	0.89	**0.99**	0.94	0.82	0.95
Pconv	0.83	0.88	0.57	0.9	0.83	0.92
Ours	**0.96**	**0.92**	0.96	**0.96**	**0.94**	**0.97**

Table 2: Results: Similarity in object co-occurrence scores (higher is better) for our method vs the baselines on Cityscapes dataset. We calculated SOCC scores for all class labels in the dataset. Out of the 34 different class labels in the dataset, we picked 5 random pairs for display purposes. For most of them, our method achieves better SOCC scores compared to baselines.

the cityscapes dataset. It can be seen that our method most closely resemble the ground truth. Our methd not extrapolates the existing objects ensuring texture and structural continuity but is capable of adding very precise novel objects. However, all the other baselines generally are unable to add any new objects and struggle with ensuring continuity.

Similarly in figure 6, we show visual comparisons of our method with the baselines for ADE20K bedroom dataset. Our method clearly is able to add novel objects in each of the images shown, while the baselines merely extrapolate the existing objects.

4.2. Quantitative performance

Table 1 shows the FID scores and Table 2 shows the Similarity in object co-occurrence of our method compared to the baselines on the two datasets. Note that all these scores are on the validation split of the two datasets. We outperform all the baselines by very significant margins. Our method obtains 20 and 50 points improvements over the nearest baselines on Cityscapes and ADE20K respectively.

Table 2 shows the SOCC scores for different pairs of object classes in the cityscapes dataset. It can be seen that our method is able to generate results that consistently resemble the object co-occurrence statistics for most class pairs in the dataset.

4.3. Effect of instance maps

We do ablation study on the proposed method to analyze the effect of instance maps in our final stage. To this we train our own baseline method wherein the image generator takes only the extrapolated label map as input. Figure 7 shows the visual comparison between the proposed method that uses instance maps and the baseline without instance maps (and hence also the feature maps). We show that the generated instance maps aid in generating crisp boundaries between instances of the same class and also in generating feature maps using the IaCN module. This shows the superiority of our approach over the baseline that does not take into account the instance maps.

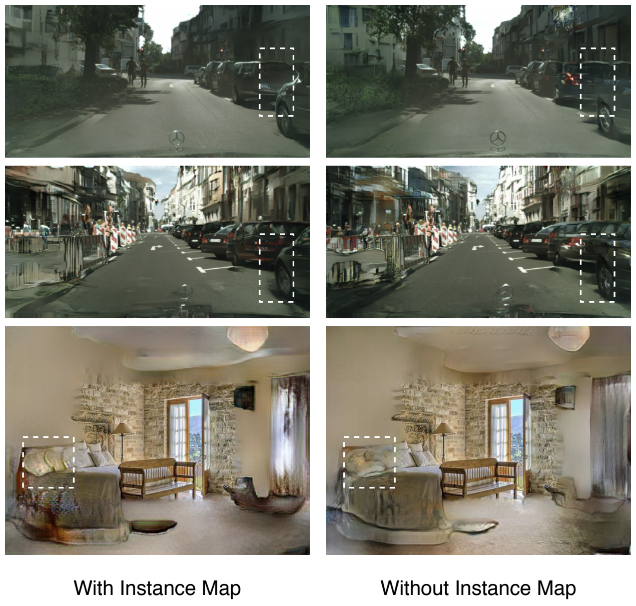

Figure 7: Our method w/ instance maps produce better object distinction of objects in the white dotted marked region in the outpainted image(cars in the first and second row, pillows in the third row).

5. Discussion and Conclusion

We propose a new solution for image extrapolation that is amenable for adding novel objects as well extending the existing objects and textures. Our solution distinguishes itself from all previous works in the image extrapolation by extrapolating the image in semantic label space. We show in the paper that this helps us achieve our objective of adding new objects. We also propose the generation of panoptic label maps from just segmentation maps, which enables us to create multiple instances of the same classes and as well allow us to have control over the instances thus created. We show in our supplementary video how our method can be recursively applied to generate image extrapolations of arbitrary dimensions. We hope our work encourages researchers to develop solutions for image editing in semantic label space.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2

[2] Samaneh Azadi, Michael Tschannen, Eric Tzeng, Sylvain Gelly, Trevor Darrell, and Mario Lucic. Semantic bottleneck scene generation. *arXiv preprint arXiv:1911.11357*, 2019. 3

[3] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009. 2

[4] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020. 3, 4

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 5

[6] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016. 4

[7] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016. 2

[8] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346, 2001. 2

[9] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1033–1038 vol.2, 1999. 2

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2

[11] 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. In *Advances in neural information processing systems*, pages 6626–6637, 2017. 6

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2, 4

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 4

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 5

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 5

[17] Kuldeep Kulkarni, Tejas Gokhale, Rajhans Singh, Pavan Turaga, and Aswin Sankaranarayanan. Halluci-net: Scene completion by exploiting object co-occurrence relationships. *arXiv preprint arXiv:2004.08614*, 2020. 3, 5

[18] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019. 6

[19] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 4

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3

[21] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018. 2, 6, 7

[22] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *Advances in Neural Information Processing Systems*, pages 570–580, 2019. 2, 4

[23] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017. 3

[24] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*, 2018. 2

[25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 3, 4

[26] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 2, 3, 4, 5, 6

[27] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33, 2020. 5

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[29] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020. 3

[30] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10521–10530, 2019. 2, 6, 7

[31] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 2, 3, 4

[32] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1399–1408, 2019. 2, 4, 6, 7

[33] Yaxiong Wang, Yunchao Wei, Xueming Qian, Li Zhu, and Yi Yang. Sketch-guided scenery image outpainting. *arXiv preprint arXiv:2006.09788*, 2020. 2

[34] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8456–8465, 2018. 5

[35] Zongxin Yang, Jian Dong, Ping Liu, Yi Yang, and Shuicheng Yan. Very long natural scenery image prediction by outpainting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10561–10570, 2019. 2

[36] Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[37] Changqian Yu, Jingbo Wang, Changxin Gao, Gang Yu, Chunhua Shen, and Nong Sang. Context prior for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12416–12425, 2020. 3

[38] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018. 2

[39] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019. 4

[40] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 3

[41] Xiaofeng Zhang, Feng Chen, Cailing Wang, Ming Tao, and Guo-Ping Jiang. Sienet: Siamese expansion network for image extrapolation. *IEEE Signal Processing Letters*, 27:1590–1594, 2020. 2

[42] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 3, 5

[43] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 5

Semantically-Aware Image Extrapolation – Supplementary material

Anonymous CVPR 2021 submission

Paper ID 9939

1. Network Architectures

The stage-1 uses PSPnet [11] to obtain the semantic label maps from the input images. The generators used in stage-2 (semantic extrapolation) and stage-4 (image generator) are inspired from SPADE [6] generator and consist of 9 spade residual blocks. The multiscale Discriminators used in stage-2 and stage-4 are similar to that of pix2pixHD's [9]. The encoder of stage-4 (that forms VAE [4] with the generator) is inspired from the encoder used in SPADE [6]. The co-occurrence patch discriminator of stage-4 is inspired from the one used in swapping autoEncoders [7]. Our stage-3 which is used to convert the semantic label map into class-agnostic center predictors and center off-sets is a pure-generator network. For this stage, again our network architecture is similar to stage-1 and stage-4.

Instance-aware Context Normalization (IaCN)
This module takes in input the cropped RGB image and the instance map. The instance map is used to get the partial instances. Partial instances refer to the instances which are part of the input image and need to be completed in the final out-painted image. The mean colors for all these instances are calculated using the input cropped image and then, the instance-aware context normalization feature map or the output of the IaCN module is obtained by putting these mean colors in the extrapolated part of the instances in the outside region. Figure 1 shows some of the input-output pairs for IaCN module.

2. Panoptic Map generation (stage-3)

The stage-2 of our pipeline extrapolates the semantic label maps. However, in order to differentiate between multiple instances of the same class, we need to generate the panoptic label map from the thus extrapolated semantic label maps. In a typical panoptic label segmentation set-up, we have access to the full image. Instead, in our case, what we have access to the full semantic label maps obtained as outputs from stage-2. To obtain the panoptic label maps, we take inspiration from [1] that converts the image into two parallel branches 1. semantic label maps and 2. pixel-wise instance center maps and x and y off-set maps from the instance centers. The center maps and the off-set maps thus predicted are used in conjunction with the semantic label maps to obtain the final panoptic label map. For the panoptic segmentation branch, Cheng et al. [1] obtain class-agnostic instance centers and off-sets from the instance center for every location. Class-agnostic centers refer to center locations for the different instances that belong to the 'things' categories. In addition, for every pixel that belongs to the 'things' categories, we define the x-offsets and y-offsets as δx and δy, respectively, of that pixel location from the center of the instance the pixel belongs to. Here, instead of having the above mentioned two parallel branches, we train a network to obtain the center maps and the x and y offset maps from the semantically extrapolated label maps that are the outputs of stage-2. The ground-truth center maps are represented by Gaussian blobs of standard deviation of 8 pixels, centered at the instance centers. We use a simple L_2 loss to compute the instance center loss and L_1 loss to compute the offset losses. The final loss for stage-3 is the weighted sum of the center loss and the offset losses.

During the test time, we adapt the procedure mentioned by [1] to group the pixels based on the predicted centers and off-sets to form instance masks. The instance masks and the semantic label map (the input to stage-3) are combined by majority voting to obtain the panoptic label map. For more details, the readers are encouraged to refer to [1]. The thus obtaine panoptic label map is used in stage-4 for our instance-aware context normalization as well as to obtain the instance boundary maps.

3. Detailed Training and Testing algorithms

The detailed training algorithms for stage-2 and stage-4 are given in Algorithm 1 and 2

In Algorithm 1, we use the Ground Truth segmentation map Y', Ground Truth Instance Map X and Cropped Segmentation L as the inputs. We obtain a segmentation map, L_1, of desired resolution by zero padding it at the periphery. An instance Boundary map, B_1 is created from X. Y is generated using the SPADE[6] generator which has an extra output channel (apart from the input classes) for the bound-

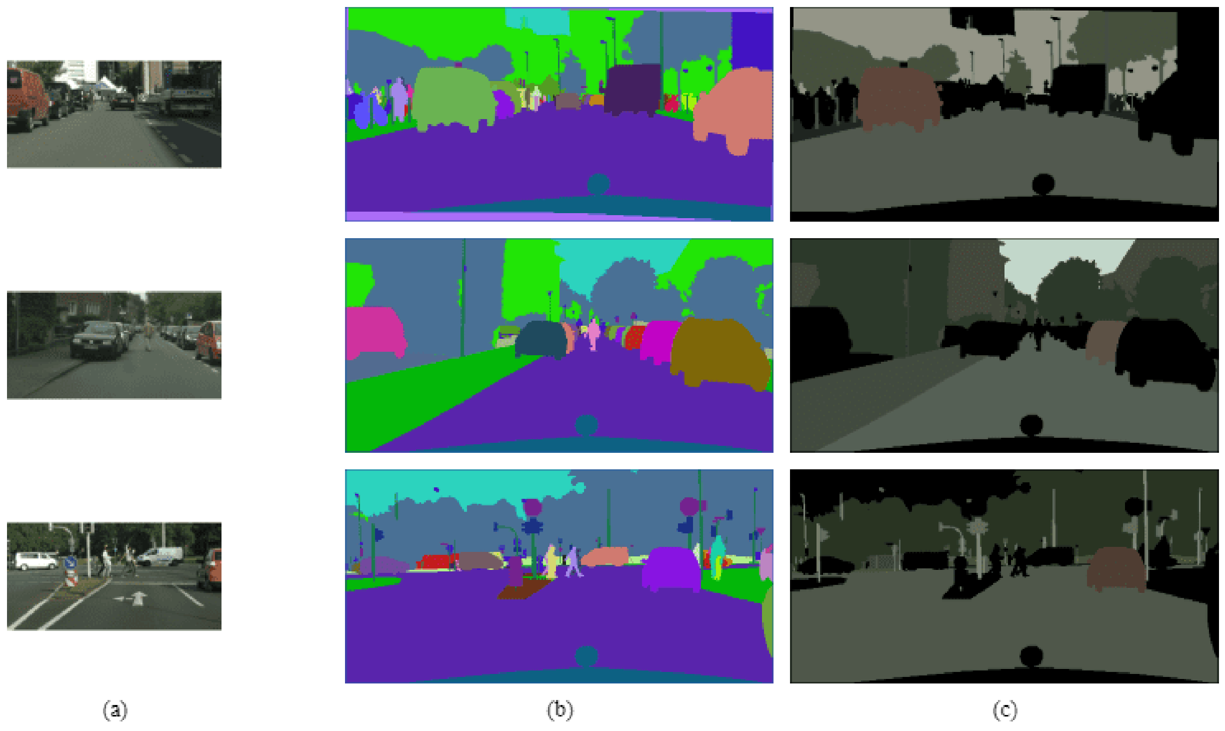

Figure 1: Input-Ouput for IaCN: (a) Input cropped image (b) Instance map for ground truth (c) Feature map obtained by IaCN. IaCN computes the mean colors for all the partial instances from the cropped image and pastes them at the location of those respective instances. For all the other instances, it outputs black.

ary map. We use an additional cross entropy loss between this extra channel of Y and B_1. The Multiscale Discriminator, $D_{\text{multiscale}}$ distinguishes between the generated segmentation map (Y) and the ground truth segmentation map (Y'). The model tries to minimize the train objective function for semantic label map extrapolation (Equation 1 in the main paper). Finally, the parameters of G, $D_{\text{multiscale}}$ are updated accordingly.

In Algorithm 2, we obtain X' by concatenating Ground Truth segmentation map (L), input cropped image (X), boundary map obtained from ground truth instance map and the feature map obtained using Instance-aware Context Normalization (IaCN) module. The out-painted image (Y) is generated using generator G, which takes in X' and the encoded input image ($E(X)$). The Multiscale Discriminator $D_{\text{multiscale}}$ tries to distinguish between the generated image (Y) and the ground truth image (Z'). The Patch Discriminator (D_{patch}) tries to distinguish between fake patch ($crop(Y)$) and real patch ($crop(Z')$) obtained from the fake image and real images respectively. For each pair of fake patch and real patch, the patch discriminator takes 4 reference patches ($crops(Z')$) from the real image. We take a total of 4 fake patch, real patch and reference patches combination for one image. All the patches are of size

Algorithm 1: Training algorithm for stage-2

```
  Input:
  Ground Truth Seg Map: $Y' \in \{0,1\}^{2h \times 2w \times c}$,
  Ground Truth Instance Map: $X \in 2h \times 2w \times 1$,
  Cropped Segmentation Map: $L \in \{0,1\}^{h \times w \times c}$
1 Generate $L_1 \in \{0,1\}^{2h \times 2w \times c}$ by zero-padding $L$ at
    the periphery
2 Generate Boundary Map
    $B_1 \longleftarrow GetBoundary(Y)$
3 for epoch in maxEpochs do
4     $Y \longleftarrow G(L_1)$
5     $D_{\text{multiscale}}$ distinguishes between $Y$ and $Y'$
6     Minimize the objective function (Eqn. 1 in the
        main paper)
7     Update the parameters of $G, D_{\text{multiscale}}$
8 return G
```

64×64. We, then, minimize the final objective function (Equation 3 in the main paper) to update the parameters of G, E, $D_{\text{multiscale}}$ and D_{patch}.

The detailed testing algorithm is given in Algorithm 3

Method	FID
Ours w/o D_{patch}	48.72
Ours w/o IaCN	47.76
Ours	**47.67**

Table 1: FID scores (lower is better) for our method with and without IaCN and D_{patch} on Cityscapes dataset.

Algorithm 2: Training algorithm for stage-4

```
Input:
Cropped Image: $X \in \mathbb{R}^{h\times w\times 3}$,
Ground Truth: $Z' \in \mathbb{R}^{2h\times 2w\times 3}$,
Ground Truth Seg Map: $L \in \{0,1\}^{2h\times 2w\times c}$,
Ground Truth Instance Map: $I \in \mathbb{R}^{2h\times 2w\times 1}$
1 $X' \longleftarrow L \oplus X \oplus GetBoundary(I) \oplus IaCN(X, I)$
2 for epoch in maxEpochs do
3     $Y \longleftarrow G(X', E(X))$
4     $D_{\text{multiscale}}$ distinguishes between $Y$ and $Z'$
5     $D_{\text{patch}}$ distinguishes between $crop(Y)$ and $crop(Z')$, taking $crops(Z')$ as the reference patches
6     Minimize the objective function (Eqn. 3 in the main paper)
7     Update the parameters of $G, E, D_{\text{multiscale}}, D_{\text{patch}}$
8 return E, G
```

Algorithm 3: Testing algorithm

```
Input: Cropped Image: $X \in \mathbb{R}^{h\times w\times 3}$
Output: Outpainted Image: $Y \in \mathbb{R}^{2h\times 2w\times 3}$
1 $L \longleftarrow PSPNet(X)$                 // Stage-1
2
3 $L_1 \longleftarrow stage2(L)$               // Stage-2
4
5 $I \longleftarrow PanopticLabelMap(L_1)$     // Stage-3
6
7 $X' \longleftarrow L \oplus X \oplus GetBoundary(I) \oplus IaCN(X, I)$
8 $Y \longleftarrow stage4(X')$
```

4. Additional Implementation details

We trained and tested our model on Cityscapes [2] and ADE-20K bedroom subset [12] datasets. For stage-2 we used a batch size of 8, while for stage-4 we used a batch size of 16. All the experiments were run on 4 16GB V100 GPUs. Both the datasets were trained for 200 epochs. We used the TTUR [3] update rule.

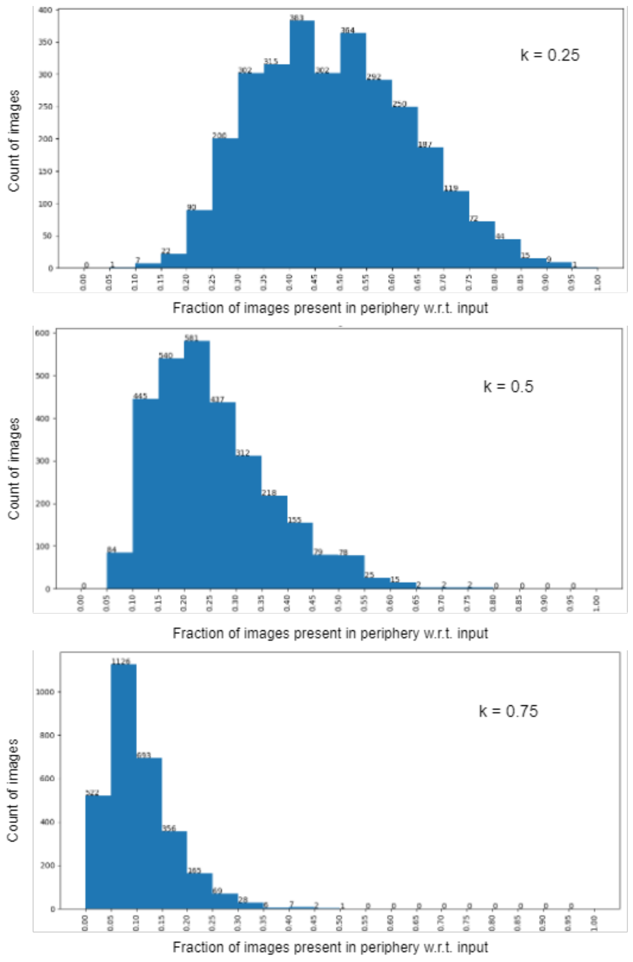

Figure 2: Fraction of images present in periphery is the ratio of number of instances outside the input crop with that inside the input crop. For $k = 0.25$, the peak occurs around 50%, for $k = 0.5$, the peak occurs around 25% and for $k = 0.75$, the peak occurs around 10%.

5. Different crop analysis

We analyzed different crop sizes. For given images $X_0 (\in \mathbb{R}^{h\times w\times c})$ in the dataset, we tried 3 different crop ratios $k = \{0.25, 0.5, 0.75\}$. We, thus, obtain the cropped images $X (\in \mathbb{R}^{k.h\times k.w\times c})$. We calculated the number of instances outside the crop region with respect to those inside. If the number of instances outside is high compared to those inside, the training information decreases, while if the number of instances outside is low compared to the input crop, the number of new instances to be learned by the model decreases. After analyzing the results (Figure 2), we chose an optimal value of $k = 0.5$.

Method	(Bed, Lamp)	(Wall, Window)	(Bed, Curtain)	(Floor, Table)	(Wall, Painting)
Outpainting-SRN	0.66	0.82	**0.94**	0.77	0.64
Boundless	0.79	0.82	0.87	0.75	0.76
Pconv	0.75	0.85	0.83	0.77	0.83
Ours	**0.82**	**0.90**	0.84	**0.87**	**0.84**

Table 2: SOCC scores (greater is better) on ADE20K bedroom subset dataset. The baselines include results from Partial Convolutions (PConv) [5], Boundless [8] and OutPainting-SRN [10]

6. Infinite Zoom

Our model has ideally an infinite zooming potential. We can zoom out an image infinitely by recursively passing the image through our model, without the model breaking upto a good extent. We experimented this by training on modified Cityscapes dataset, made by removing the Mercedes at the bottom and then upsampling the images to the original dimension. We did this since if we ”infinitely” zoomed the original images, then there would have been Mercedes recursively at the bottom in the zoomed out image. The images were recursively passed 4 times to our model. In every pass, it is zoomed out by 4 times, so we zoomed the given image 64 times. The Supplementary video contains the zooming-out video.

7. Human Evaluation

We further compare our method with the baselines via human subjective study. The baselines included results from Partial Convolutions (PConv), Boundless and OutPainting-SRN. About hundred random people were given a set of 20 randomly selected images from the Cityscapes and the ADE20K-bedroom dataset. They were given unlimited time to make the selection. They were asked to rank the images each of the image based on two parameters, viz. Realistic appearance and New object generation.
Graph 3 shows the evaluation results. We found that our results were strongly favoured by all the people on both the datasets.

8. Additional Quantitative study

Table 2 shows the Similarity in Object Co-occurrence (SOCC) scores for ours and baselines on ADE20K bedroom subset dataset.

9. Additional Ablation study

9.1. Use of extra boundary map in stage-2

We justify using extra boundary map for the training of stage-2. This extra boundary map channel helps in creating instances with better shapes since the model now knows about the boundaries of each instance. Figure 4 shows the visual comparison between the semantic label maps obtained from stage-2 with and without using boundary map channels.

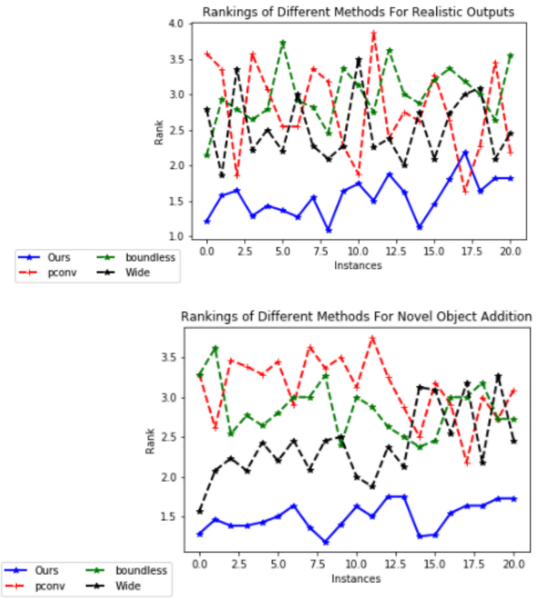

Figure 3: User preference study on the rank of various baselines. Lower rank means better.

9.2. Importance of IaCN and D_{patch}

We studied the importance of Instance-aware Context Normalization module and Co-occurrence Patch discriminator in our pipeline on Cityscapes dataset. Table 1 shows the FID scores for the three methods. Though there isn't much FID difference between with and without IaCN but there is a significant visual difference between the two (Figure 5).

9.3. Failure of single stage approaches

All the baseline methods are based on direct image-to-image translation in a single stage. To analyse the strength of our approach and the failure of single stage approaches,

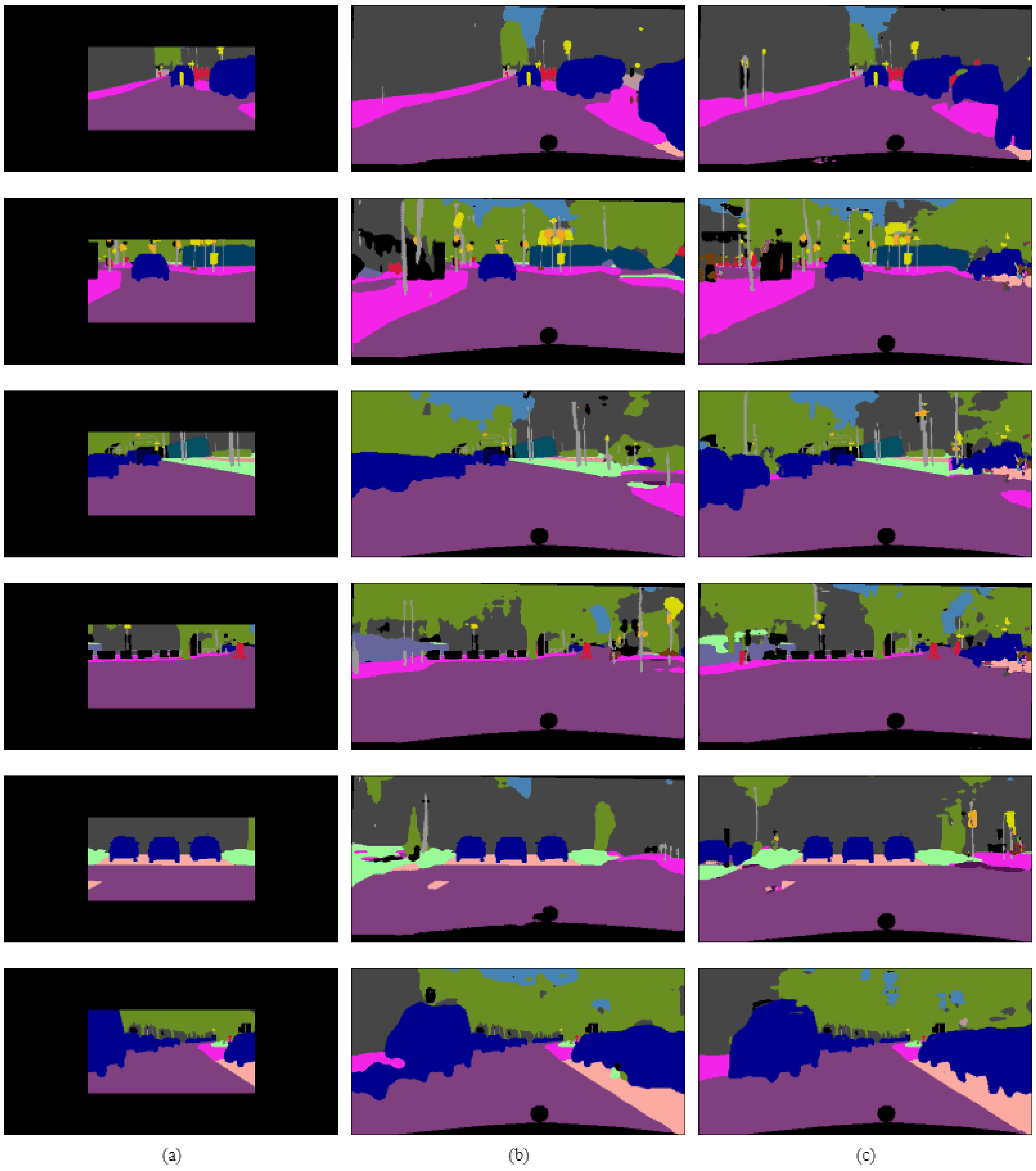

Figure 4: Visual comparison between with and without extra boundary map concatenation in the ground truth for the stage-two training: (a) Input segmentation map (b) The resulting segmentation map without using the extra boundary map channel (c) The resulting segmentation map with the boundary map channel.

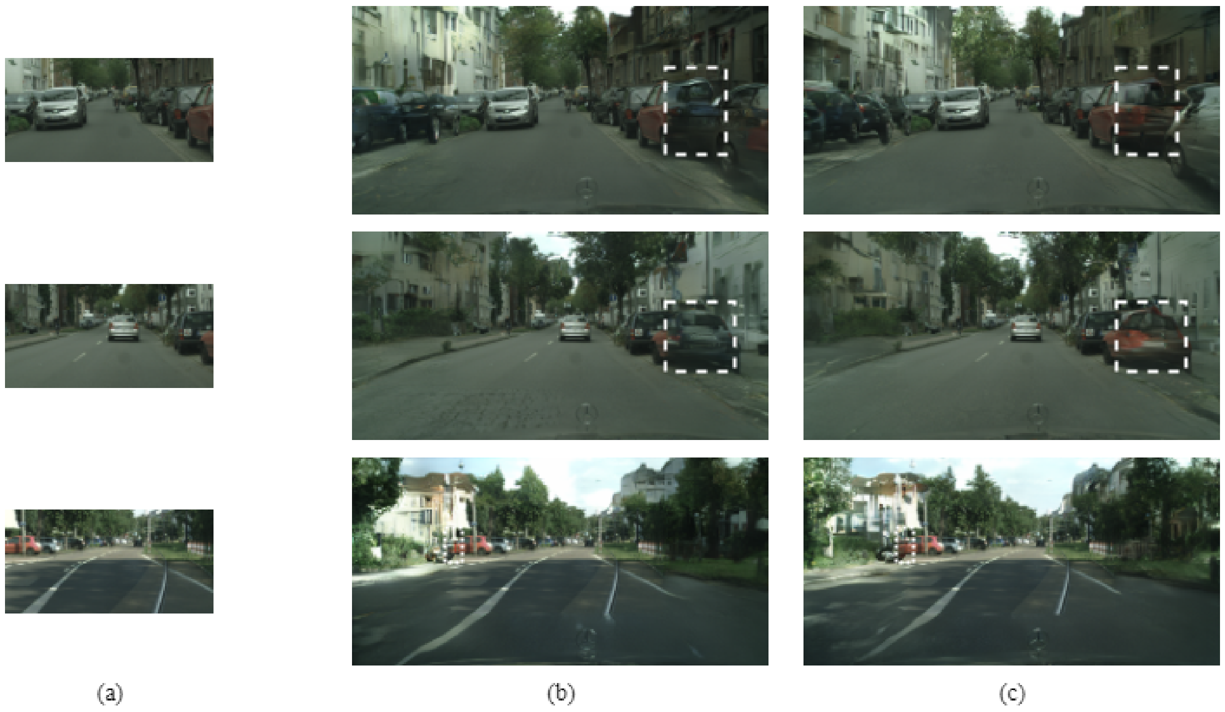

Figure 5: Visual comparison between with and without IaCN: (a) Input cropped image (b) Output image without IaCN (c) Output image with IaCN. The mean colors of the partial instances are transferred to the final extrapolated instances in the case of IaCN.

we try direct image extrapolation by image-to-image translation. For this, during the training time, we used the RGB cropped input image and extrapolated it to the final RGB image using the stage-4 (Instance-aware image synthesis stage), without IaCN and D_{patch}, directly (Figure 6). We observed that the images are generated with mere texture extension at the periphery with minimal to no new object generation.

10. More results and comparisons

Figures 7, 8, 9 shows some more comparison of our method with the baselines.

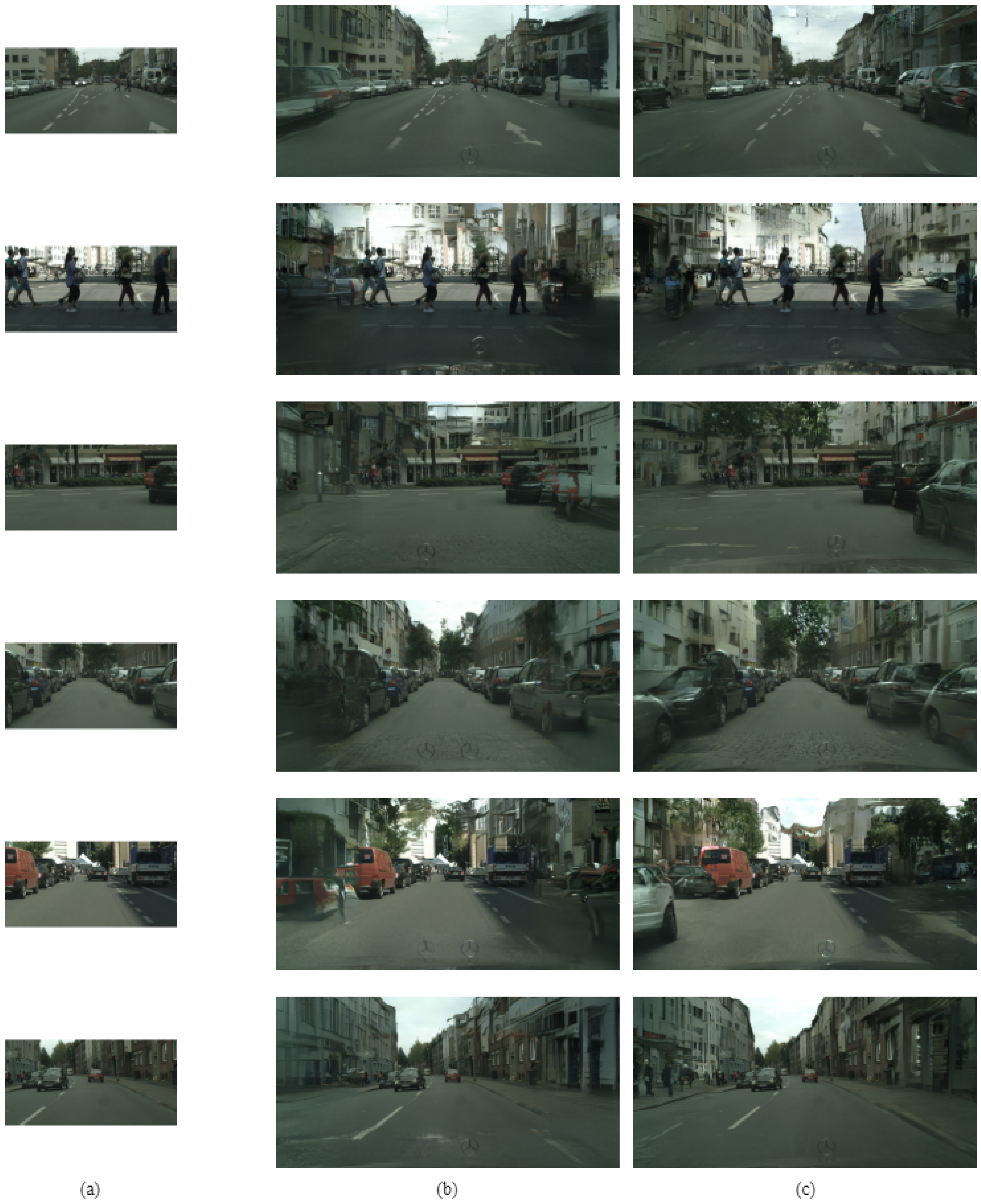

Figure 6: Comparison between ours and single stage approach: (a) Input cropped image (b) Extrapolated image using our single stage approach (c) Extrapolated image using our current approach.

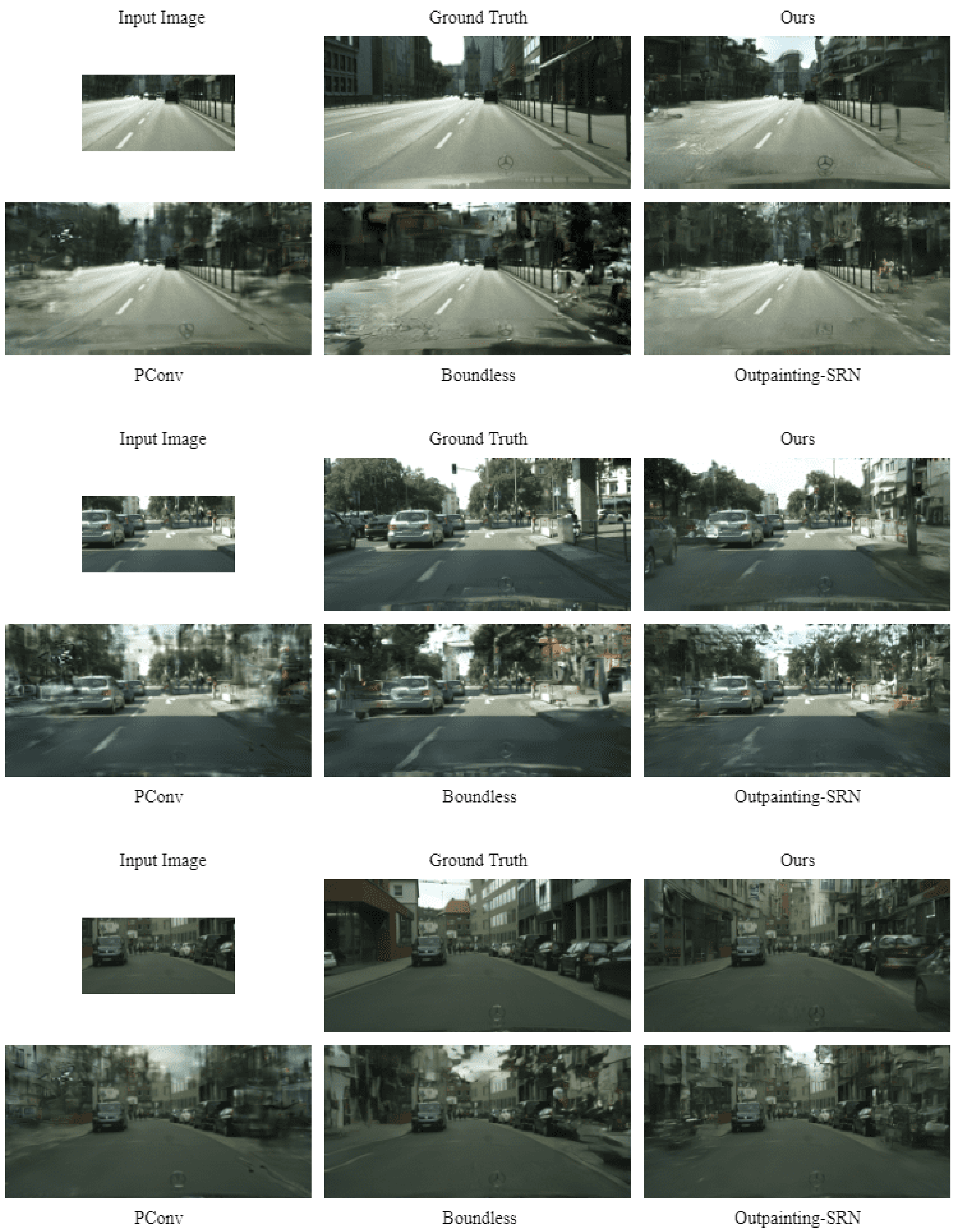

Figure 7: Visual comparison between our model and the baselines on Cityscapes dataset. The baselines include results from Partial Convolutions (PConv) [5], Boundless [8] and OutPainting-SRN [10]

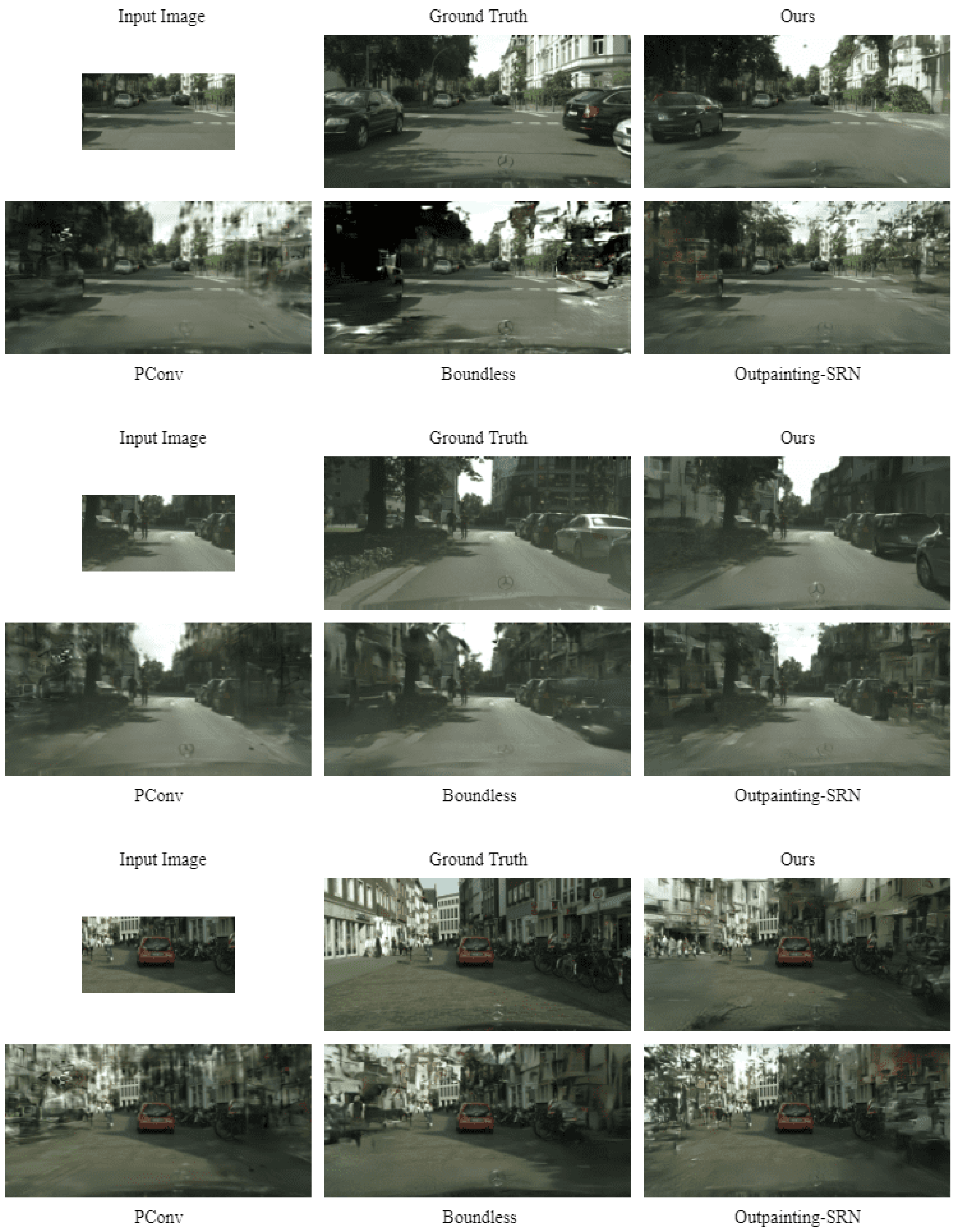

Figure 8: Visual comparison between our model and the baselines on Cityscapes dataset. The baselines include results from Partial Convolutions (PConv) [5], Boundless [8] and OutPainting-SRN [10]

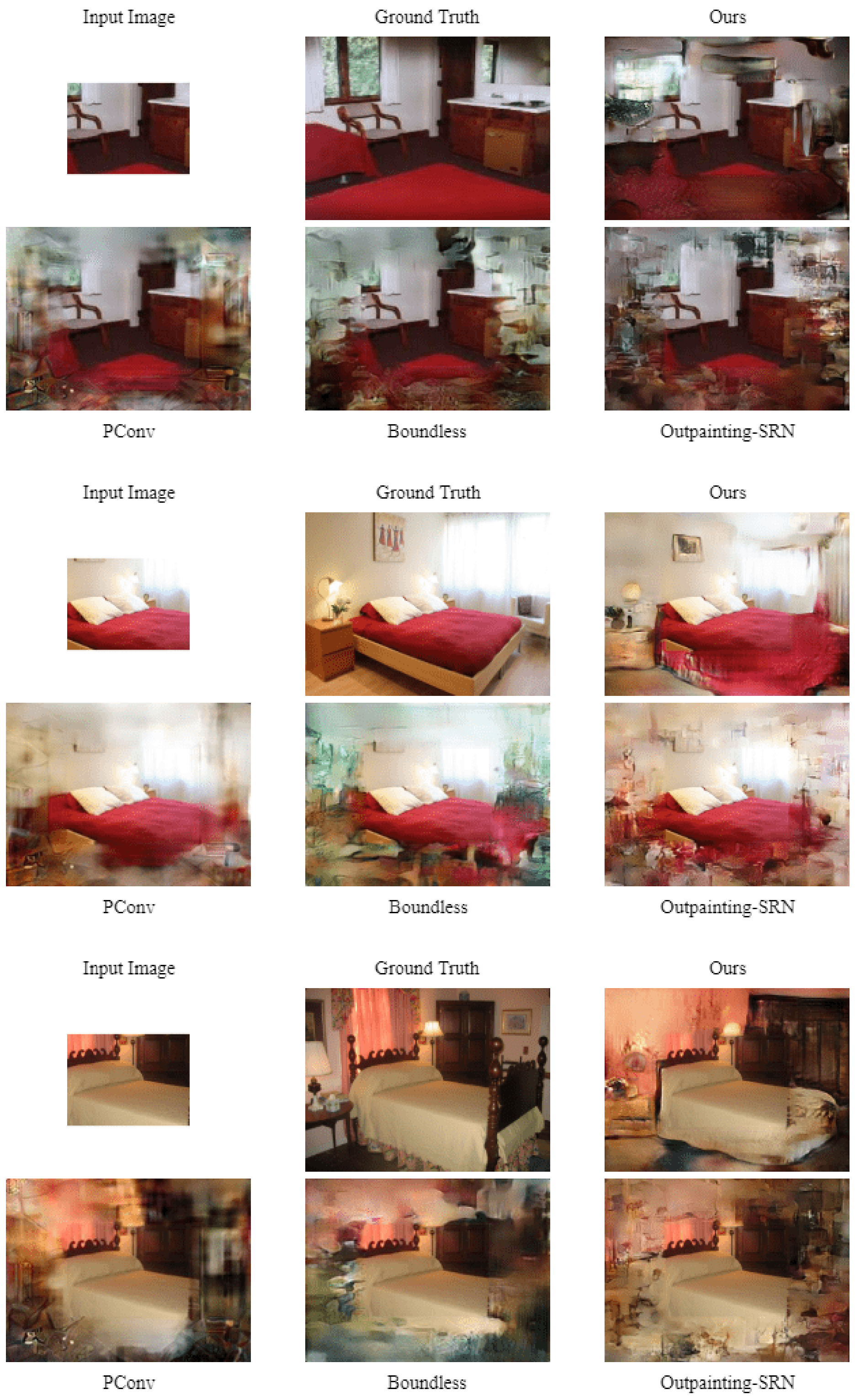

Figure 9: Visual comparison between our model and the baselines on ADE20K-bedroom dataset. The baselines include results from Partial Convolutions (PConv) [5], Boundless [8] and OutPainting-SRN [10]

References

[1] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020. 1

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 3

[3] 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. In *Advances in neural information processing systems*, pages 6626–6637, 2017. 3

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[5] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018. 4, 8, 9, 10

[6] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 1

[7] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[8] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10521–10530, 2019. 4, 8, 9, 10

[9] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 1

[10] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1399–1408, 2019. 4, 8, 9, 10

[11] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 1

[12] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3