



Seamless Satellite-image Synthesis

Jialin Zhu  and Tom Kelly 

University of Leeds

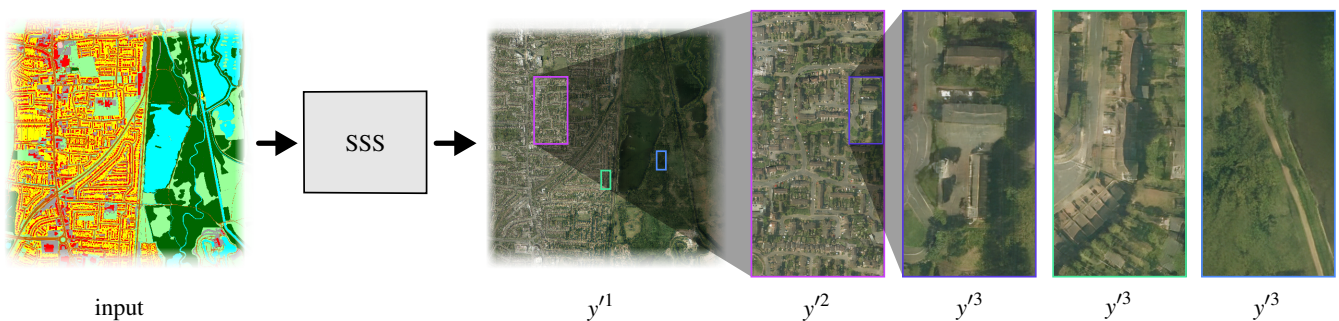


Figure 1: Our system, SSS, takes input vector cartographic data (left), and generates realistic and seamless high-resolution satellite images (right; the image y^3 is 8,192 pixels wide, approximately 67 megapixels) which are continuous over space and scale. Given the input, SSS generates massive spatially seamless images over a range of scales (y^1 , y^2 , and y^3) by blending the results of lower resolution networks.

Abstract

We introduce *Seamless Satellite-image Synthesis (SSS)*, a novel neural architecture to create scale-and-space continuous satellite textures from cartographic data. While 2D map data is cheap and easily synthesized, accurate satellite imagery is expensive and often unavailable or out of date. Our approach generates seamless textures over arbitrarily large spatial extents which are consistent through scale-space. To overcome tile size limitations in image-to-image translation approaches, SSS learns to remove seams between tiled images in a semantically meaningful manner. Scale-space continuity is achieved by a hierarchy of networks conditioned on style and cartographic data. Our qualitative and quantitative evaluations show that our system improves over the state-of-the-art in several key areas. We show applications to texturing procedurally generation maps and interactive satellite image manipulation.

CCS Concepts

• *Computing methodologies* → *Texturing; Image processing;*

1. Introduction

Satellite images have revolutionized the way we visualise our environment. Massive, high-resolution, satellite images are essential to many modern endeavors including realistic virtual environments and regional- or city-planning. However, these images can be prohibitively expensive, out of date, or of low quality. Further, continuous satellite images covering large areas are often unavailable due to limited sensor resolution or orbital mechanics. In contrast, cartographic (map) information is frequently accessible and available over massive continuous scales; further, it can be easily manually edited or synthesized. In this paper we address the problem of generating satellite images from map data at arbitrarily large resolutions.

Convolutional neural networks are state of the art for image to image translation; such networks can translate from cartographic images to satellite images. However, these networks are unable to generate outputs of arbitrary areas – the resolution of these synthetic satellite image tiles remains limited by available memory and compute. Arranging many such small tiles to cover a large map results in inconsistencies (seams) between the tiles. Further, if a larger number of tiles are joined together and viewed at multiple scales, the result lacks continuity and style coordination.

Inspired by convolutional image-to-image translation systems such as Pix2Pix [IZZE17] and SPADE [PLWZ19], *Seamless Satellite-image Synthesis (SSS)*, Figure 1) creates satellite image tiles by conditioning generation on cartographic data. These tiles

are of fixed resolution (256×256 pixels in our implementation) and exhibit good continuity internally, but would contain spatial and scale artifacts if naïvely tiled over a larger area. *Spatial discontinuities* between adjacent tiles are caused by a lack of coordination between network evaluations, resulting in lack of continuity for properties not determined by the cartographic input (e.g., color of building roof or species of tree). We introduce a network which removes such spatial discontinuities by leveraging learned semantic domain knowledge as well as the cartographic data. *Scale discontinuities* between a larger set of synthetic satellite tiles arise from unnatural style variations over larger areas (e.g., distribution of cars throughout a city or different camera sensors). To address this, SSS uses a hierarchy of networks synchronized by color guidance.

The SSS system contains three components – two convolutional neural networks to generate tiles of different resolutions (*map2sat*), and remove seams (*seam2cont*) between those tiles, as well as an interface allowing the generated textures to be explored interactively. SSS contributes:

- A neural approach to generate massive scale-and-space continuous images conditioned on cartographic data.
- A novel architecture to train and evaluate image-to-image generation networks at high resolutions with reduced memory use.
- An application to texture procedurally generated cartographic maps – allowing the the synthesis of textures for entirely novel areas following user style guidance.
- An application to interactive exploration of endless satellite images generated on-the-fly from cartographic data.

The textures generated by SSS have many applications, including interactive city planning, virtual environments, and procedural modeling. During city planning, the textures may help stakeholders (e.g., homeowners, planners, and motorists) understand proposed developments. For example, sit-down sessions in which stakeholders interactively edit a map and are shown birds-eye-view (satellite) “photos” within seconds. Further, virtual environments may be constructed using SSS’s output (e.g., a flight simulator) to texture massive terrains on-the-fly, or the streets and open areas of urban procedural models [PM01].

In the remainder of this work we will discuss the corpus surrounding our work in Section 2, introduce our system in Section 3, give the details of the neural architectures and their coordination in section 4 before evaluating our results qualitatively and quantitatively in section 5. Source code and weights for our system are available online at <https://github.com/Misaliet/Seamless-Satellite-image-Synthesis>.

2. Related Work

Texture synthesis is the generation of texture images from exemplars or lower dimension representations. We refer the reader to Wei et al. [WLKT09] for an overview. Non-neural approaches from the AI Winter create images with local and stationary properties using techniques such as Markov Random Fields to generate pixels [EL99, WL00] or patches [LLX*01, EF01] from an exemplar image. Variations on these themes generate larger, more varied textures [DSB*12]. Procedural languages for texturing [MZWVG07]

are well adapted for urban domains and can be used to guide synthesis [GAD*20].

With the advent of **Convolutional Neural Networks (CNNs)**, texture synthesis has increasingly become data-driven, creating images by gradient descent in feature space [GEB15] or with Generative Adversarial Networks (GANs) [ZZB*18]. Another approach to texture generation is to transform an input image to a different style or domain while retaining the structure. Such *image-to-image* translation is implemented by the influential UNet architecture [RFB15] which passes the input through an “hourglass” shaped network of convolutional layers. This allows the network to aggregate and coordinate features within a *bottleneck*. UNets have been applied to generate satellite images [IZZE17] from map data, however the resolution of the output tiles was limited by available memory. Image-to-image networks typically disentangle the structure from style to enable style transfer [GEB16]. These have been applied to generate terrains [GDG*17] and within chained sequences [KGS*18] for architectural texture pipelines.

Label-to-image translation is a variation of image-to-image translation; the input is a label-map with each pixel having a specific class, such as cartographic data containing, streets, buildings, and vegetation. Again, we note the application of hourglass networks [WLZ*18]. We build on Semantic Image Synthesis with Spatially-Adaptive Normalization (SPADE) [PLWZ19] which uses a simpler V-shaped architecture with a bottleneck at the start, rather than center, of the network. From a latent code, successive *SPADE Residual Blocks* generate feature-maps of increasing resolution. Within these blocks, a SPADE unit introduces learned per-label style information using Spatially-Adaptive Normalization.

Texture blending or inpainting approaches give a way to combine multiple smaller images into a larger one. These may match small image patches [LLX*01, EF01, BSFG09], or minimize a function over a grid of pixels using seam energy minimization [AS07, KSE*03, RKB04], exemplars [CPT03], or a Poisson formulation [PGB03]. Although such local approaches produce excellent pixel-level results, they do not contain the higher, semantic-level knowledge to blend textures and images with complex structures. We may overcome this limitation with user input [ADA*04], large datasets [HE07], and distributions learned by neural architectures [ZZL19, ZWS20, BSP*19]. These learn distributions for the blended area using techniques such as guided Poisson constraints [WZZH19] and symmetry [DCX*18]. Domain-specific image blending has been studied for domains such as microscopy [LCH*15], façade texturing [KGS*18], and satellite generation. TileGAN [FAW19] searches a library of latent encodings of adjacent micro-tiles to create seamless images, rather than learning a domain specific blending function.

In contrast to relatively recent satellite images, applications of **cartography** date back to antiquity. Today, there are high-quality, publicly available maps of much of the world [Fou21, Ser21]. We may also generate maps with a variety of techniques, including procedural systems; see Vanegas et al. [VAW*10] for an overview. In particular, we may use rule and parameter controlled geometric algorithms to recreate the distribution of streets [CEW*08], buildings [PM01], and city parcels [VKW*12]. These procedural systems can generate massive quantities of 3D city geometry from

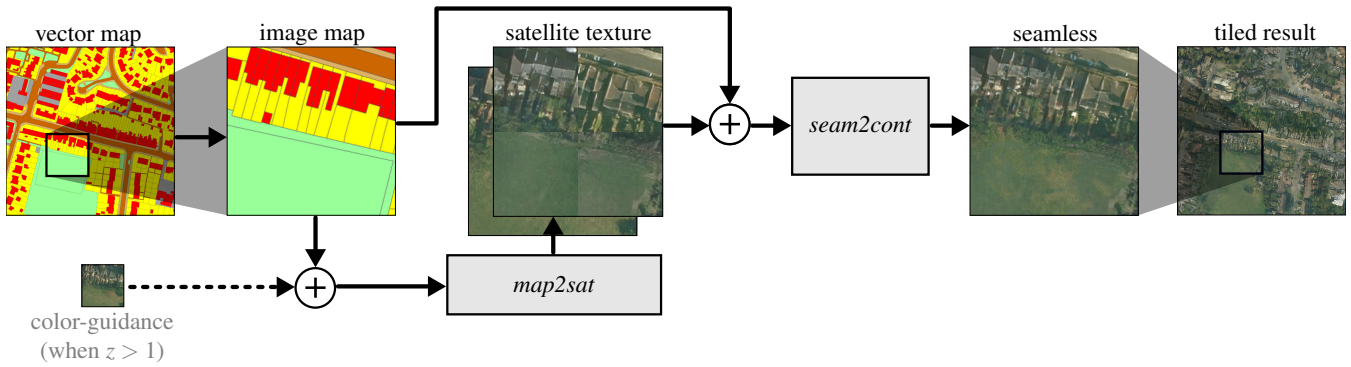


Figure 2: SSS uses a hierarchy of networks for different scales. A pair of networks is shown above, which process input cartographic maps to large seamless images at a single scale level, z . First, a vector map of arbitrary extent is processed to create an image tile of map data. The *map2sat* network then performs a translation to the satellite texture domain, before the *seam2cont* network removes visible tiling artifacts to create a seamless image and the result is combined with neighboring tiles to create an image of arbitrary resolution (right). Subsequent, smaller scale, *map2sat* networks use previous outputs as scale-guidance (bottom left).

random seeds; such geometry can be used to generate cartographic maps by projecting the 3D meshes to 2D images and applying a constant color for each category (e.g., red buildings). Procedurally generated cartography has the advantage that it can be easily edited with a user-interface or created from databases.

3. Overview

We propose a novel method for map-to-satellite-image translation over arbitrarily large areas. Two neural networks work together to first generate satellite image tiles from map data, and then remove seams and discontinuities from these images. To generate an area, SSS (Figure 2) first generates two overlapping sets of satellite image tiles with the *map2sat* network. Then the *seam2cont* network removes the discontinuities and seams between adjacent tiles by learning to generate a mask to blend the overlapping satellite images. Joining together such tiles results in a single seamless image with a large resolution and scale.

To achieve our goals, we solve problems such as the training of a mask generator to remove seams while realistically blending tiles, the coordination of networks over different areas to create continuous results in scale-space, and reducing the memory usage for large image synthesis networks.

4. Method

SSS uses convolution neural networks (CNNs) to learn to synthesize satellite image by learning over datasets. We prepare a dataset of paired map and satellite images for each tile at a fixed resolution of 256×256 pixels. Vector cartographic data is processed to image tiles – each GeoPackage [geo] file contains a list of polygons, each with one of 13 semantic class labels (e.g., *building*, *road* or *track*, or *natural environment*). The polygons are ordered canonically and rendered into an image, assigning every a pixel a label. The corresponding satellite images have 8-bit color depth, a resolution 0.25 meters per pixel, and metadata containing their location and extent in the local Coordinate Reference System; we use this transform to align them to the map data.

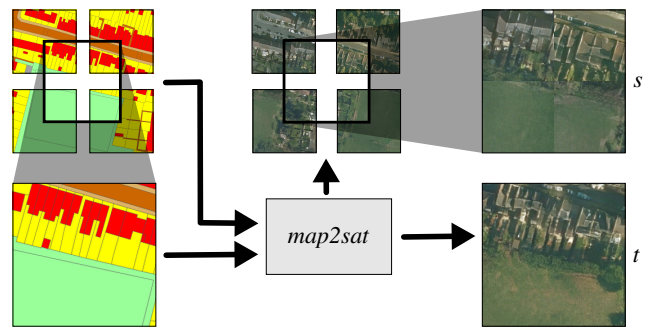


Figure 3: Creation of *seam2cont* input tiles. We obtain one image with seams, s and one without seams, t , from the *map2sat* network run over map data (left). Note that the corners of s are seamless against adjacent s tiles as they come from the same *map2sat* execution.

The paired dataset is used to train our image-to-image translation network, *map2sat*, based on the SPADE [PLWZ19] architecture. The network learns to translate our semantic cartographic label image tiles into corresponding satellite image tiles. However, to synthesize massive areas two problems must be solved.

The first problem is maintaining a realistic style over different scales of tiled areas: *scale-space continuity*. If our tiles are too large, we are able to synthesize a large area, but the limited fixed resolution provides poor detail (see additional materials *variants_8k_images/large_scale_only.jpg*). On the other hand, if the tiles are too small, we see unrealistic variation over the tiled image – this may be too little variation, resulting in a homogeneous image (e.g., very similar trees over an entire city); see additional materials *variants_8k_images/small_scale_only.jpg* – or too much variation, leading to a chaotic swings in its variation between adjacent tiles (e.g., roads suddenly changing color). Additionally, our interactive system allows users to explore a dynamically generated satellite image at a range of scales; in this application, it is neces-

sary to generate imagery at a given scale without overly onerous number of generator network applications. SSS uses a scale-space hierarchy of generator networks to create realistic variation over areas at multiple scales (Section 4.1). By evaluating the hierarchy tiles lazily, the interactive system is able to quickly evaluate tiles at arbitrary scales.

The second problem is that simple tiling of the synthetic satellite images to cover a large area does not produce natural images, but creates an image with seams; state-of-the-art CNN architectures do not provide *spatial-continuity* between adjacent tiles. This is largely because of their reliance on spatially narrow bottleneck layers, which allow tile-wide coordination of features at the cost of limiting maximum tile size. Adding additional layers to increase the tile size becomes prohibitively expensive due to the memory requirements of these large layers. Another source of seams is the padding [AKM*21] around layer boundaries, where synthetic values (i.e., zero or flipped) are used to pad convolution input; these introduce seam artifacts around the perimeter of network outputs. We address this spatial-continuity problem in the following section.

4.1. Spatial Continuity: *seam2cont* Network

The limited resolution of CNN image synthesis techniques, such as those we use in the *map2sat* network, leads to seam discontinuities when we join multiple tiles together. Here we introduce the *seam2cont* network which removes the seams, allowing tiling over an arbitrarily large area. Given tiles generated by our image-to-image network we introduce a novel architecture to blend pixels from these existing tiles to create a seam-free image. We found blending (masking) a sufficient approach to seam removal as the tile content is well aligned by strong cartographic conditioning, and blending over a wide area allows the network to identify semantic plausible locations for each transition.

For the area of one tile i , we use two independent, overlapping layers of *map2sat* evaluation to synthesize tiles s_i and t_i . The areas of s_i and t_i are offset so for each tile i , there is a seam-free option, as Figure 3. Our goal is to blend s_i and t_i such that the result has no internal seams (as t_i), but also has no seams with neighboring tiles (as s_i). The quarters of s_i are continuous across the tile boundary because they have come from the same output tile of a single *map2sat* evaluation. Thus, the problem of removing seams is transformed to a masking (image blending) problem – where the mask predominantly selects pixels from t_i in the center and from s_i in the periphery. In-between it should select pixels which maximize realism. We propose a method to learn to generate the mask, m , which selects how to blend s_i and t_i . By learning a mask generator, *seam2cont* can learn the semantic meanings of the inputs, producing higher quality results than optimizing for gradient [KSE*03] or search [FAW19].

For one tile area i , *seam2cont* takes the map image x_i (including label and instance data), the generated satellite image with seams (s_i), and the generated satellite image without seams (t_i). The output of the *seam2cont* generator is a mask, m_i , which is used to blend the input satellite images. The output of the system is seamlessly tillable satellite image, y'_i , with a feature distribution similar to the seamless ground-truth images.

Figure 4 shows the architecture of the *seam2cont* network. We use a UNet [RFB15, ZZZP*17] as our generator M to create a mask, m_i , from inputs x_i , s_i and t_i :

$$m_i = M(x_i, s_i, t_i) \quad (1)$$

To evaluate the quality of the generated mask, an output tile, y'_i is blended from s_i and t_i by B :

$$y'_i = B(s_i, t_i, m_i) = s_i * m_i + t_i * (1 - m_i) \quad (2)$$

M is a 8 layer UNet which we train with three goals for the blended image y'_i : (1) tileable – with s_i near the borders, (2) realistic, and (3) internally seamless. We construct a differentiable loss function to achieve these three goals which can be backpropogated to improve M during training.

To address goal (1) the mask should select s_i at the edge of the tile to ensure continuity with adjacent tiles. We implement this by forcing the outside part of generated mask m_i to be white (to have a value of 1) in F , guided by a constant circular mask c (Figure 4):

$$m'_i = F(m_i) = m_i * (1 - c) + c \quad (3)$$

and we can modify equation 2 to give us the masked output:

$$y'_i = B(s_i, t_i, m'_i) = s_i * m'_i + t_i * (1 - m'_i) \quad (4)$$

To support goal (2) we use a discriminator $D_{realism}$, a PatchGAN [IZZE17], to reward M for the realism of the masked output. $D_{realism}$ is trained to discriminate between generated images and those from the ground-truth (without seams). When training M , it takes results y'_i and the constant circular mask c . These conditions allow the discriminator to locate the output seams [TSK*19] and increase accuracy. The loss $\mathcal{L}_{realism}$ is then:

$$\mathcal{L}_{realism} = \mathbb{E}_{x_i, s_i, t_i} [-D_{realism}(y'_i, c)] \quad (5)$$

Following the standard GAN architecture, this loss using $D_{realism}$ is used during training to update M to increase realism.

Goal (3) is to remove internal seams from y'_i . To achieve this, we train an image-to-image CNN to locate seams and use it as a discriminator, D_{seams} . It encourages M to generate a mask m_i which removes seams from the blended result. It is an image-to-image translation network which learns to identify seams in patches of y'_i , outputting a grayscale image that is white (value of 1) where there is no seam and black (value of 0) where there is a seam. D_{seams} is used as an adversarial loss when training M by constructing a L2 loss between the discriminator's output and the desired (seamless; value of 1) result. Following CUT [PAEEZ20], we extract random patches, P , from the output of mask to contribute to the loss:

$$\mathcal{L}_{seams} = ||D_{seams}(P(y'_i)) - 1|| \quad (6)$$

Finally, as Pix2Pix [IZZE17], we add a L1 loss to the ground-truth in order to improve the training stability:

$$\mathcal{L}_{recons} = |y'_i - y_i| \quad (7)$$

The total loss function that is propagated through M during training is then:

$$\mathcal{L}_M = \lambda_1 \mathcal{L}_{realism} + \lambda_2 \mathcal{L}_{seams} + \lambda_3 \mathcal{L}_{recons} \quad (8)$$

We train the model with Adam [KB15] with a learning rate of

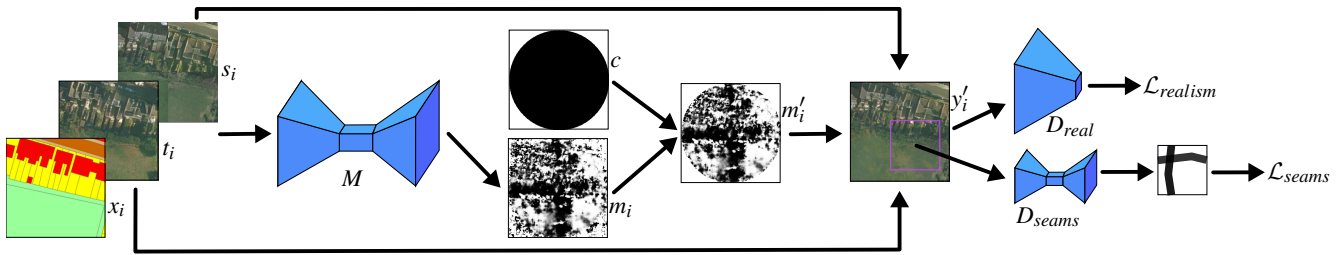


Figure 4: The data-flow when testing the seam2cont network (black arrows). The inputs are map label images x_i , as well as generated satellite images s_i and t_i . M generates a mask, m_i which is multiplied by a fixed circle image c to create m'_i . This final mask blends s_i and t_i to create our spatially continuous seamless image y'_i . Cyan arrows: the two discriminators contribute losses when training M .

0.0002. We may write the complete *seam2cont* network, S , which generates creating seamless image, y'_i , as:

$$S(x_i, s_i, t_i) = B(s_i, t_i, F(M(x_i, s_i, t_i))) = y'_i \quad (9)$$

We train the neural networks M , $D_{realism}$, and D_{seams} together, performing one SGD update for each network at every iteration. $D_{realism}$ is trained to discriminate between blended result images (y'_i) and image without seams (t_i):

$$\mathcal{L}_{d-realism} = \mathbb{E}_{x_i, s_i, t_i} [D_{realism}(s_i, c) + (1 - D_{realism}(y'_i, c))] \quad (10)$$

D_{seams} is trained to identify seam locations in images. We find that the simplicity of this image based seams/no-seams formulation complements $D_{realism}$, which works on more complex image space features. We use random patches with D_{seams} as they are smaller and faster to train. Random patches also have non-constant output (with seams in different locations) as they are unaffected by boundary layer padding. We train D_{seams} to discriminate between the output n and a seams mask image showing the position of seams, l , with the loss:

$$\mathcal{L}_{d-seams} = ||D_{seams}(P(y'_i)) - P(l)|| \quad (11)$$

where P is a particular random crop of resolution 64×64 pixels.

The complete loss function of *seam2cont* network that is back-propagated through M when training is:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{realism} + \lambda_2 \mathcal{L}_{seams} + \lambda_3 \mathcal{L}_{recons} + \lambda_4 \mathcal{L}_{d-realism} + \lambda_5 \mathcal{L}_{d-seams} \quad (12)$$

The training data for the whole *seam2cont* network contains seams images and non-seams images, both of them are generated by *map2sat* at different scales. We train *seam2cont* network with a dataset of around 3,000 images on a Nvidia Titan V GPU for 200 epochs using hyperparameters $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, $\lambda_3 = 10.0$, $\lambda_4 = 1.0$, and $\lambda_5 = 1.0$.

The primary resolution of all networks in SSS is 256×256 pixels. Larger resolutions are possible, but would require significantly increased memory. This limited resolution allows us to study impact of seam-removal with reasonable compute resources.

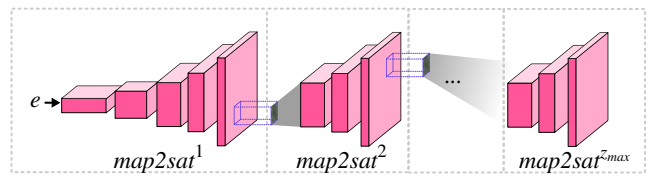


Figure 5: The *map2sat* sub-networks can be trained independently or end-to-end. This figure illustrates the organization of SPADE residual blocks (magenta, Figure 6). *map2sat*¹ is a SPADE network taking latent input e . The output of *map2sat*¹ is cropped (purple) to provide color guidance to the *map2sat*² sub-network. *map2sat*² and later sub-networks have the same architecture.

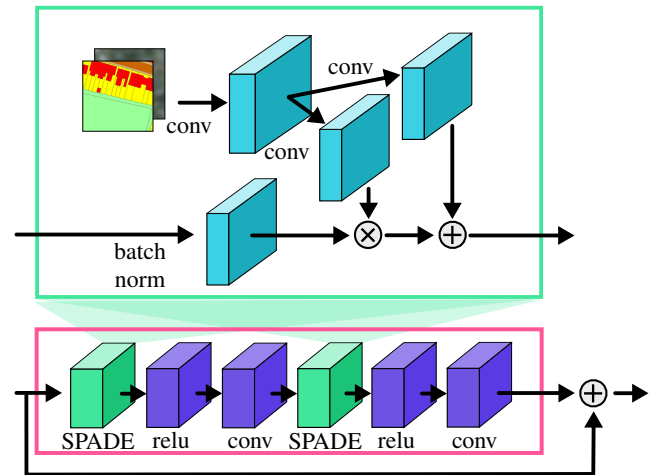


Figure 6: Top (green): Extending the original SPADE block [PLWZ19], we concatenate the RGB color guidance from the previous *map2sat* scale and the map image. The concatenated tensor is used to learn convolutions to manipulate the channels after batch-normalisation. Bottom: SPADE residual blocks are combined with ReLU and convolutional layers to create the magenta blocks shown in Figure 5.

4.2. Scale-Space Continuity: *map2sat* Network

The *seam2cont* network allows us to build large composite images from generated tiles. To generate these tiles, we build on SPADE [PLWZ19] to create realistic individual tiles from map label images. These tiles can be assembled into an arbitrarily large image. However, there is no coordination between tiles which are far apart. To address this scale-space problem, we introduce the *map2sat* network to generate synthetic image tiles with scale-space continuity.

To provide consistency at different scales, we would ideally use a single large tile, however such high resolution is prohibitively computationally expensive. Instead, *map2sat* uses multiple sub-networks to generate tiles at different scales, retaining a fixed resolution of 256×256 pixels. For example, at the largest scale, $z = 1$, map image tiles are 1 kilometer wide (3.9 meters per pixel), while at a smaller scale, $z = 2$, a map tile is four times larger: 250 meters wide (0.98 meters per pixel). Using multiple scales allows large scale tiles to provide scale-space synchronization and small scale tiles to provide high detail; however, they must be consistent.

In the case of our interactive exploration system, we can achieve the necessary speed by limiting the number of tile evaluations to balance speed against scale or detail. For example, we do not need to generate small scale tiles if the user is viewing a large area. However, small and large scales must be consistent – the same areas must have the same content and style. To achieve this scale-space continuity, each *map2sat* sub-network is conditioned on a guidance subsection of the larger scale tile.

The input to *map2sat* at each scale is a map label image and an optional 3-channel color guidance image which is only applied when scale level $z > 1$. The color guidance image is from a larger scale in our hierarchy satellite image that provides color conditioning to ensure consistency between scales. The output of each *map2sat^z* sub-network is a RGB satellite image consistent with both the map images and the larger-scale color guidance (if provided). The map image, color guidance, and output satellite image have the same resolution.

The principle issue when adapting a CNN generator architecture to an arbitrary resolution is avoiding the memory requirements of large layers. Using the *seam2cont* network we can refactor the network into sub-networks, *map2sat^z*, for each scale $z \in 1 \dots z^{max}$ (see Figure 5). By cropping each sub-network's input, the memory and data requirements are reduced considerably. With this cropped architecture we are able to train the whole network end-to-end when required, but each sub-network remains independent when evaluated. This sub-network independence allows portions of scale-space to be evaluated to accelerate our interactive editor.

To implement this refactoring, a spatial crop from 256×256 pixels to 64×64 pixels is inserted between each adjacent sub-network (see Figure 5); this provides a $4 \times$ decrease in scale per sub-network. A 3×3 convolution layer to 512 channels is positioned at the start of each sub-network to preserve the original SPADE network architecture as much as possible – we found this arrangement did not impact generation quality.

The inputs to each sub-network are map label images with optional color guidance images from the previous (larger) scale for

sub-networks $z > 1$. The color guidance images coordinate style and content consistency between sub-networks at scales *map2satⁿ* and *map2sat⁽ⁿ⁺¹⁾*. Experimentally, we observe that the SPADE network takes the majority of its guidance from the label image which overpowers the spatial convolutional input guidance. To compensate, we append additional color guidance to the semantic label channel in the spatially-adaptive denormalization blocks (Figure 6). We found that this approach better preserves color guidance information. The color guidance image is blurred with a Gaussian filter ($\sigma = 8$ at 256×256 pixels) to only keep the color information and reduce other high frequency features. The spatial input to the largest scale *map2sat¹* at training time is from an encoder that is trained to create a latent encoding, e , from a satellite image; at test time we use a normally distributed e to create variety over our results.

The complete *map2sat* network is a streamlined version of a high-resolution SPADE with considerably lower memory usage. For instance, the on-disk weights of a full SPADE network trained at 256×256 pixels are 436.9MB while the weights for a cropped sub-network, *map2sat^{z>1}*, are 31.5MB. For a $8,192 \times 8,192$ pixels image the full SPADE GPU memory requirement exceeds 32GB, while the factored weights for the two networks in the SSS system are less than 12GB.

Real data is used to train each of the *map2sat* sub-networks independently. End-to-end fine-tuning over the whole network results distributions were sufficiently close to realistic data that at test-time these networks would accept synthetic inputs from the previous *map2sat* when $z > 1$. The training data of sub-network *map2satⁿ* is semantic label map images and color guidance images generated by scale level *map2satⁿ⁻¹*. We use around 3,000 images to train the sub-networks. We implement our pipeline with z^{max} of 3 and train every *map2sat^z* network for 70 epochs.

4.3. Complete SSS Pipeline

Our entire system is a combination of the *seam2cont* and *map2sat* networks (Figure 2) over the scale levels. The architecture is shown in Figure 7 and the tile geometry in Figure 8. At the z^1 scale level, we use *map2sat¹* with random latent encoding e , but without color guidance. The satellite images tiles s_i^1 and t_i^1 are generated and processed by the *seam2cont¹* network for spatial continuity, resulting in y_i^1 . The set of seamless tiles, y_*^1 , are stitched together to create the final image at scale z^1 ; we crop this result to use as the color guidance of z^2 . This process continues over all scales $z \in 1 \dots z^{max}$ to generate satellite images of different scales over an area.

Because our method generates tiles and stitches them together to create huge images, we can generate satellite images of arbitrary size from cartographic map data. Furthermore, we can enforce scale continuity over a wide range of scales. Another advantage of our SSS pipeline is that the two independent network architectures and independent networks at each scale can be used for custom applications; for example, the z^2 *seam2cont* network may be used to remove seams in real-world satellite images, or we may insert additional sub-modules to create satellite images with smaller scales.

The general complexity of the entire SSS pipeline depends on the scale level, $z = k$, and the linear scale factor, f , used between each sub-network. For example, a scale factor of $f = 4$ between each

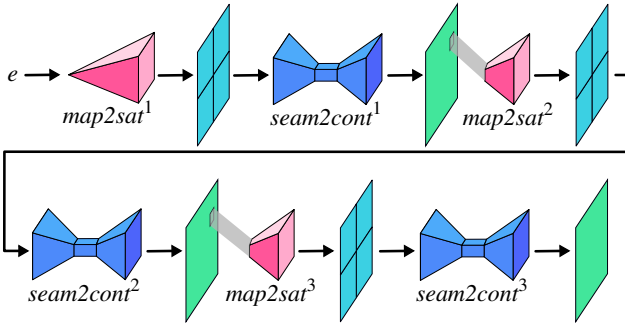


Figure 7: SSS combines a series of *map2sat* and *seam2cont* networks. Here we show a 3-scale ($z^{max} = 3$) pipeline. *map2sat* sub-networks generate satellite image tiles which are processed by the *seam2cont* networks to remove spatial discontinuities.

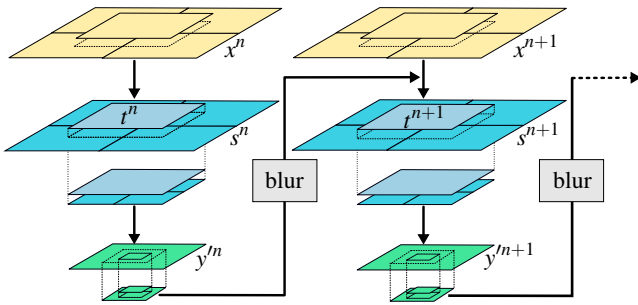


Figure 8: The geometry of SSS tiles. Left, right columns: network outputs at scales $z = n$ and $z = n + 1$. Scale-space continuity is provided by passing a blurred crop of y^n (green bottom left) to $map2sat^{n+1}$ (cyan mid-right).

sub-network implies that the smaller scale sub-network has $f^2 = 16$ tiles covering the area of a single larger scale tile. To generate one scale, all larger scales must also be generated. The worst-case number of *map2sat* evaluations, assuming we generate 4 tiles at $z = k$ is:

$$\sum_{i=1}^k 4/(f^2)^{i-1} \tag{13}$$

and worst-case number of *seam2cont* evaluations is:

$$\sum_{i=1}^k (\sqrt{4/(f^2)^{i-1}} - 1)^2 \tag{14}$$

5. Results

In the following results we use SSS with $z^{max} = 3$ and a scale factor of $f = 4$ to create a 512×512 pixels image at z^1 , a $2,048 \times 2,048$ pixels image at z^2 , and a $8,192 \times 8,192$ pixels image at z^3 .

Figure 9 shows the output of our technique for real-world map guidance images; further results are given in our additional materials (see folder *sss_8k_images/*). Figure 10 shows the output of

our technique for synthetic map labels generated using a procedural modeling system. Figure 11 illustrates the style continuity of SSS pipeline.

Finally, our interactive system is introduced in figure 12. We allow users to move around in the map, zoom in or out, and generate corresponding satellite images of different styles. We also provide some operations to modify the map, such as randomly replacing a single tile in the map input. All networks (*map2sat* and *seam2cont*) for 3 scales can be loaded into the 12GB memory of a Nvidia Titan V graphics card. Generation times for a 512×512 pixels window vary between 2.4 and 3.42 seconds depending on the scale. Smaller scales require additional network evaluations.

6. Evaluation

6.1. Methodology

Evaluating the quality of composite synthesized images for neural networks is a difficult problem; there is no canonical method to assess the performance of image stitching. Traditional edge detection technology, such as Sobel filter detection and Canny edge detection, can be used for seams detection but are not optimized for tile-based systems where the seam locations can be anticipated. In addition, these algorithms output images with edges rather than comparable values.

To evaluate our *seam2cont* network results, we use two methods. First, we design a qualitative user study survey. We ask users to identify the best system for removal of seams between tiles to examine the effectiveness of our *seam2cont* network against other algorithms. Second, we introduce a novel algorithm, MoT (Mean over Tiles), to measure the ability to remove seams in images. We evaluate against the following baselines: traditional image blending with a soft mask, Graphcut [KSE*03], Generative Image Inpainting [YLY*19], and TileGAN [FAW19].

For traditional image blending, we use s as input and blend s with t by a soft mask shown in the right bottom corner of Figure 13 (c) to obtain an image without seams. Graphcut removes seams by splicing the center part of t into s selecting a cut through a buffer area. We use a small area, 10×10 pixels, shown in the right bottom corner of Figure 13 (d) as our buffer area. We train the Generative Image Inpainting model, which uses gated convolutions [YLY*19] for the image inpainting task; it is trained with our generated seams dataset s and the diamond shape mask shown in the right bottom corner of Figure 13 (e). This is to let the model learn to generate the center diamond area which covers the seams. Generative Image Inpainting does synthesize reasonable results, but it often changes the original content structure in s as it is not conditioned on map data. Last, we train PGGAN [KALL17] with ground-truth satellite images of 256×256 pixels resolution for TileGAN. TileGAN can generate large seamless images by searching the latent space of the pre-trained PGGAN model to create tiles that are stitched into a large image. It is not strictly an image-to-image translation system, but can take a small, low-resolution, image as guidance to generate large, high-resolution, images.

Figure 13 shows selected representative output tiles for each of

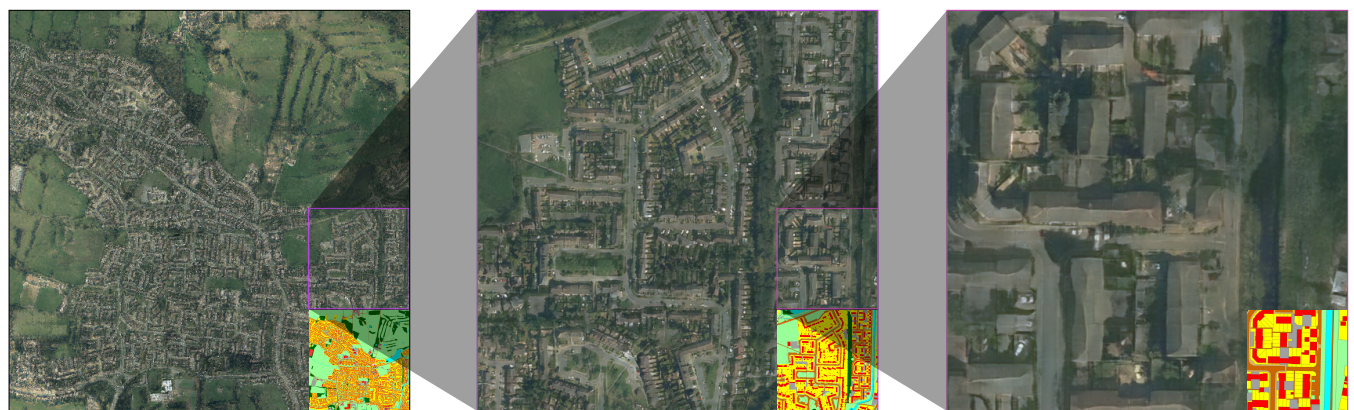


Figure 9: Results from the SSS pipeline. Rows: examples of generated satellite images from real-world map data. Columns: left to right show scales $z = 1, 2, 3$. Inset are the corresponding map inputs.



Figure 10: Result of our system on synthetic procedurally generated map data.

the baselines. Traditional image blending in Figure 13 (c) does a good job generally, but since it uses a fixed mask, the rough shape of the mask can sometimes be observed. This is more obvious when we apply image blending on green fields. Figure 13 (d) is created by the Graphcut algorithm, which clearly shows visible seams. Figure 13 (e) is generated by Generative Image Inpainting; it does syn-

thesize good results, however, it changes the original structure of Figure 13 (a); traces of the diamond mask can also be observed.

Figure 13 (f) is a tile from a large image synthesized by TileGAN; an $8,320 \times 8,320$ pixel image is guided by a low-resolution image which is cropped into the corresponding tile. TileGAN cre-

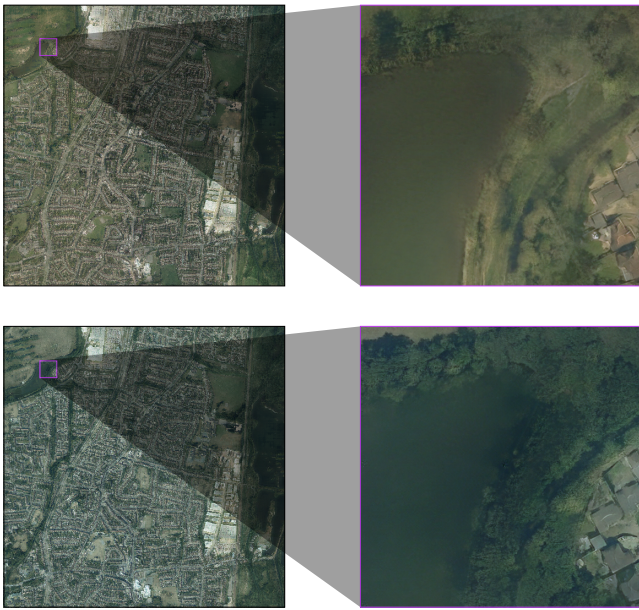


Figure 11: Style generation of z^1 (left) and z^3 (right) images in a 3 scale level SSS system. Each row has a different latent encoding, e , input to $map2sat^1$, giving different styles which remain consistent throughout the scale levels.



Figure 13: This figure illustrates results from selected baseline techniques. (a) t from our $map2sat$ network; (b) the output of our $seam2cont$ network; (b, inset) the calculated mask, m ; (c) the result of fixed-mask (c, inset) image blending. (d) the Graphcut algorithm calculates four minimum energy seams within margins (d, inset); (e) Generative Image Inpainting is not conditioned on map data, but is used to inpaint a diamond area (e, inset); (f) TileGAN conditioned on a lower-resolution satellite image.



Figure 12: User interface for our interactive system. In this figure, we show a result of generated satellite image (right) by corresponding map (left). Here the user has edited the map by rotating the central tile. Our system is able to generate realistic results even with the discontinuities in the map data.

ates an overly “blobby” result with poor large scale features. While TileGAN is true to the guidance image, the guidance is too low resolution to effectively guide the generation of crisp man-made features such as streets. When TileGAN is trained on satellite images of urban areas, the results can be poor because of the difficulty in searching for a latent encoding which matches large, sharp features over tile boundaries. This is illustrated in the additional materials over file *baselines_8k_images/tilegan.jpg*. We provide large area results (8k images) of all baselines in *baselines_8k_images/*.

We also evaluate against the following variants and later use these to form an ablation study: photographic ground-truth, ground-

truth with seams (created by flipping each tile along the diagonal), original SPADE [LZZE17] with no *seam2cont* networks, our *map2sat* results without *seam2cont* network, and our results which are generated by procedural map data.

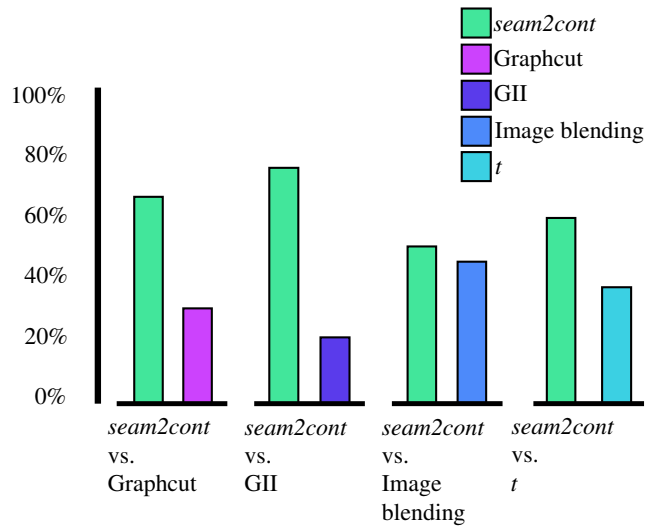


Figure 14: User study results. We show results of SSS (our approach), Graphcut, Generative Image Inpainting (GII) and t ($map2sat$ output, without $seam2cont$). In all scenarios, users rate our seam-removal as more natural than alternatives.

The User study survey evaluates user’s perceived quality of seam removal. We designed a user questionnaire which shows an image with seams, and ask which of two techniques remove the

seams in the most natural way. We compare against 4 baselines: Graphcut, Generative Image Inpainting, image blending with a soft mask, and the dataset t without seams that we use for training. Comparison trials for one baseline are performed over 10 pairs of images. We do not include TileGAN in the survey because it does not support generating satellite imagery of a specific area. We asked a total 20 participants “Which of these two images below removes the seams in the most natural way?” (see *misc/user_study.png* in additional materials). We also collected information on participant’s digital image editing experience. Among our participants, 55% process digital images every month, while the remaining 45% have no relevant experience.

Figure 14 shows the results of our user study. SSS removes seams in a more natural way than all the baselines in the majority of trials. Our advantage is particularly strong against Graphcut and Generative Image Inpainting. We believe this is because Graphcut cannot complete the stitching task well over a limited buffer area, and Generative Image Inpainting removes seams while changing the original content of the image. We found it surprising that when our results are compared with t , we still have an advantage. This may be because the tile generation near the border (t) may be less accurate, so showing more images from the center of adjacent tiles (y') may create more realistic results.

The MoT algorithm compares differences between horizontally and vertically adjacent pixels over generated images to highlight any sudden color differences at known tile boundaries.

To compute the MoT response we firstly crop the input image by removing ($tileWidth/2$) pixels from each edge. Let C_a be all pixels in columns $\{0 + a, tileWidth + a, tileWidth * 2 + a, \dots, imageWidth - tileWidth + a\}$ from a cropped image. Similarly R_a for all rows. The MoT response at pixel position, x, y , is the absolute difference of the averages of differences of adjacent pixels:

$$D(x, y) = |\overline{C_x} - \overline{C_{x+1}}| + |\overline{R_y} - \overline{R_{y+1}}| \quad (15)$$

We use $tileWidth = 256$ to match our generated tile size, $x, y \in 0 \dots tileWidth - 1$, and a $8,192 \times 8,192$ pixel image to calculate the MoT values.

Figure 15 shows the results of the MoT algorithm. The horizontal axis shows discontinuities at each pixel. The *map2sat* results show an obvious response peak at tile boundaries, while after the *seam2cont* network our results do not have this peak. This illustrates that our network has learned to remove the seams over the image. This algorithm also works for other baselines and variants; table 1 shows the absolute difference between of $max(D(x, y))$ and $mean(D(x, y))$ over x, y for the baselines and variants. When the difference value is greater than 0.002 a sharp vertical or horizontal seam is present.

6.2. Ablation Study

We use variants’ results from the previous section and some new variants to present a visual ablation study. First, we create an image of 512×512 pixels resolution at scale z^1 and upscale it with simple Lanczos interpolation to $8,192 \times 8,192$ pixels. Such an approach loses its detail, appearing very soft (see additional materials,

Model, Baseline, variant	difference value
Image blending with a soft mask	0.000341
Graphcut	0.000344
Generative Image Inpainting	0.001911
TileGAN	0.001336
Ground-truth	0.000291
Ground-truth with seams	0.003214
Original SPADE without <i>seam2cont</i>	0.002723
The <i>map2sat</i> without <i>seam2cont</i>	0.004724
The SSS whole pipeline	0.000730
The SSS whole pipeline with procedural map data	0.000845

Table 1: The difference between max and mean values from our MoT algorithm. We found that images with difference values greater than 0.002 contained obvious seams.

variants_8k_images/large_scale_only.jpg). Second, we create an 8k image only at z^3 , without color guidance, and note that this image looks very homogeneous. For example, most natural environmental areas are covered with the same color and often have similar patterns (see *variants_8k_images/small_scale_only.jpg*). Third, we generate an 8k image with *map2sat*, but no *seam2cont* network (see *variants_8k_images/no_sat2tile_network.jpg*). This gives us an image with seams, but because this image is generated using color guidance, the style of each tile also shows scale-space consistency; it is also better than a large image stitched together by the results of SPADE alone (see *baselines_8k_images/seams.jpg*).

7. Limitations

There are several limitations of our SSS pipeline. First, our *map2sat* is based on SPADE [PLWZ19] and Pix2Pix [IZZE17] which means that the quality of the generated satellite images is limited by these architectures. We also tried Progressive Growing of GANs when we implementing TileGAN for comparison, but there is still a gap between the generated satellite image and the ground-truth satellite image. Another limitation is that we only remove the linear seams between adjacent tiles. The *seam2cont* network also leaves theoretical singular point discontinuities at the midpoints of each tile edge after blending between s and t . We found this had minimal impact on the quality of seam removal in practice, but can envisage an additional network which removes such singular discontinuities. Finally, we produce strong results because SSS is conditioned on cartographic data, but compare to some baselines which can not use this data fully (e.g., TileGAN), or were not designed for this task (e.g., Graphcut).

8. Conclusion

We have proposed the SSS pipeline to synthesize large satellite images that are continuous both spatially and throughout scale-space at arbitrary size for given cartographic map data while not requiring large memory use. It also supports procedurally generated cartographic maps for entirely novel areas which allow users to create textures on demand. This is achieved by the *map2sat* network architecture that can keep style continuity in specified scale-space and *seam2cont* network architecture to splice small images seamlessly into one large image. We also implement a interactive system

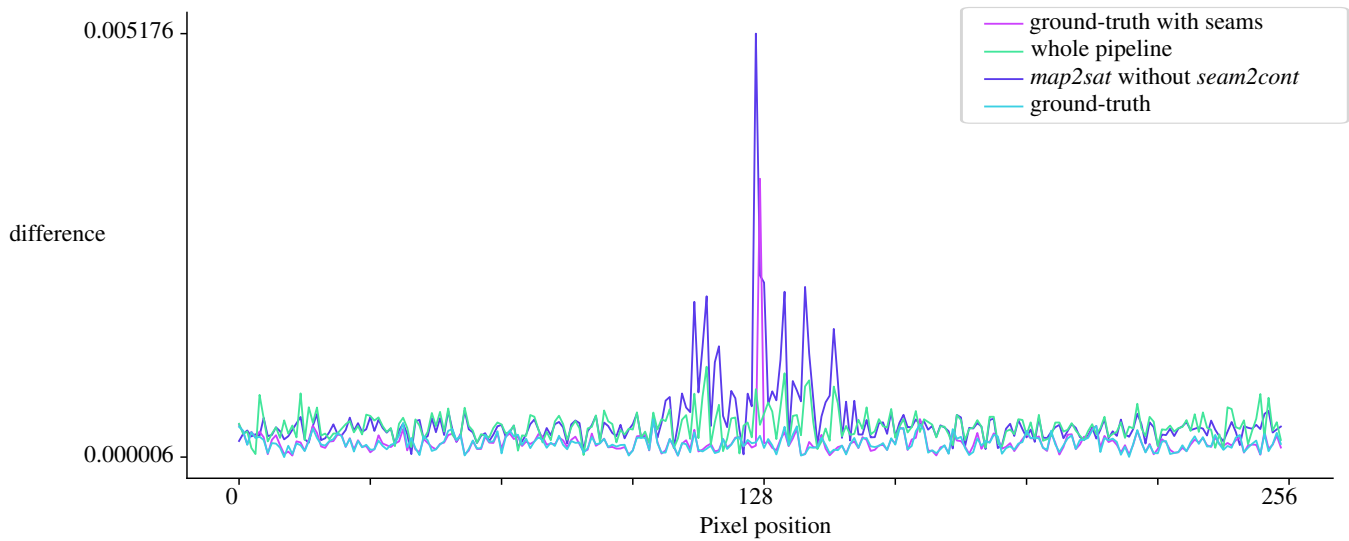


Figure 15: Selected MoT algorithm results: difference value response for ground-truth, ground-truth with seams, our map2sat with no seam2cont and our whole pipeline. Images with strong vertical and horizontal seams have a strong response on the tile boundary at 128 pixels.

to let users explore of an boundless satellite images generated conditioned on cartographic data.

In future work, we would like to study the impact of different edge constraint images (c) in our *seam2cont* network by replacing c with blob-shaped masks and using other conditioning for *map2sat*, such as multi-spectral images, pencil sketches, and low resolution images. Another avenue of exploration is the scale factor of 4 between z ; we would like to try different values with a perceptual evaluation. In addition, we suspect that using different offsets at different scales would reduce regularity and improve the quality of results. We also want to try to modify the neural network structure used; for example, adding adaptive discriminator augmentation [KAH*20]. Finally, we want to extend our work to 3D voxel based networks using transformers to synthesize a 3D virtual environment from cartographic map data.

9. Acknowledgments

We would like to thank Nvidia Corporation for hardware and Ordnance Survey Mapping [Ser21, Mas21] for map data which made this project possible. This work was undertaken on ARC4, part of the High Performance Computing facilities at the University of Leeds, UK. This work made use of the facilities of the N8 Centre of Excellence in Computationally Intensive Research (N8 CIR) provided and funded by the N8 research partnership and EPSRC (Grant No. EP/T022167/1).

References

- [ADA*04] AGARWALA A., DONTCHEVA M., AGRAWALA M., DRUCKER S., COLBURN A., CURLESS B., SALESIN D., COHEN M.: Interactive digital photomontage. In *ACM SIGGRAPH*. 2004, pp. 294–302. 2
- [AKM*21] ALSALLAKH B., KOKHLYKAN N., MIGLANI V., YUAN

J., REBLITZ-RICHARDSON O.: Mind the pad – {cnn}s can develop blind spots. In *ICLR* (2021). URL: <https://openreview.net/forum?id=m1CD7tPubNy>. 4

- [AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. In *ACM SIGGRAPH* (New York, NY, USA, 2007), Association for Computing Machinery, p. 10–es. URL: <https://doi.org/10.1145/1275808.1276390>, doi:10.1145/1275808.1276390. 2
- [BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM SIGGRAPH* 28, 3 (Aug. 2009). 2
- [BSP*19] BAU D., STROBELT H., PEEBLES W., WULFF J., ZHOU B., ZHU J.-Y., TORRALBA A.: Semantic photo manipulation with a generative image prior. *ACM TOG* 38, 4 (July 2019). URL: <https://doi.org/10.1145/3306346.3323023>, doi:10.1145/3306346.3323023. 2
- [CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. In *ACM SIGGRAPH*. 2008, pp. 1–10. 2
- [CPT03] CRIMINISI A., PEREZ P., TOYAMA K.: Object removal by exemplar-based inpainting. In *IEEE CVPR* (2003), vol. 2, IEEE, pp. II–II. 2
- [DCX*18] DENG J., CHENG S., XUE N., ZHOU Y., ZAFEIRIOU S.: Uv-gan: Adversarial facial uv map completion for pose-invariant face recognition. In *IEEE ICCV* (2018), pp. 7093–7102. 2
- [DSB*12] DARABI S., SHECHTMAN E., BARNES C., GOLDMAN D. B., SEN P.: Image melding: combining inconsistent images using patch-based synthesis. *ACM SIGGRAPH* 31, 4 (2012), 1–10. 2
- [EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 341–346. 2
- [EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *IEEE ICCV* (1999), vol. 2, IEEE, pp. 1033–1038. 2
- [FAW19] FRÜHSTÜCK A., ALHASHIM I., WONKA P.: Tilegan: synthesis of large-scale non-homogeneous textures. *ACM SIGGRAPH* 38, 4 (2019), 1–11. 2, 4, 7
- [Fou21] FOUNDATION O.: OpenStreetMap, 2021. URL: <https://www.openstreetmap.org>. 2

- [GAD*20] GUEHL P., ALLEGRE R., DISCHLER J.-M., BENES B., GALIN E.: Semi-procedural textures using point process texture basis functions. In *CGF* (2020), vol. 39, Wiley Online Library, pp. 159–171. [2](#)
- [GDG*17] GUÉRIN É., DIGNE J., GALIN É., PEYTAVIE A., WOLF C., BENES B., MARTINEZ B.: Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM TOG* 36, 6 (2017), 1–13. [2](#)
- [GEB15] GATYS L., ECKER A. S., BETHGE M.: Texture synthesis using convolutional neural networks. In *NIPS* (2015), pp. 262–270. [2](#)
- [GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2414–2423. [2](#)
- [geo] Geopackage. <https://www.geopackage.org/>. Accessed: 2010-09-30. [3](#)
- [HE07] HAYS J., EFROS A. A.: Scene completion using millions of photographs. *ACM SIGGRAPH* 26, 3 (July 2007), 4-es. URL: <https://doi.org/10.1145/1276377.1276382>, doi: [10.1145/1276377.1276382](https://doi.org/10.1145/1276377.1276382). [2](#)
- [IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. *IEEE CVPR* (2017), 1, 2, 4, 9, 10
- [KAH*20] KARRAS T., AITTALA M., HELSTEN J., LAINE S., LEHTINEN J., AILA T.: Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676* (2020). [11](#)
- [KALL17] KARRAS T., AILA T., LAINE S., LEHTINEN J.: Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017). [7](#)
- [KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *ICLR* (2015), Bengio Y., LeCun Y., (Eds.). URL: <http://arxiv.org/abs/1412.6980>. [4](#)
- [KGS*18] KELLY T., GUERRERO P., STEED A., WONKA P., MITRA N. J.: Frankengan: guided detail synthesis for building mass models using style-synchronized gans. *ACM TOG* 37, 6 (2018), 1–14. [2](#)
- [KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM SIGGRAPH* 22, 3 (2003), 277–286. [2](#), [4](#), [7](#)
- [LCH*15] LEGESSE F., CHERNAVSKAIA O., HEUKE S., BOCKLITZ T., MEYER T., POPP J., HEINTZMANN R.: Seamless stitching of tile scan microscope images. *Journal of microscopy* 258, 3 (2015), 223–232. [2](#)
- [LLX*01] LIANG L., LIU C., XU Y.-Q., GUO B., SHUM H.-Y.: Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.* 20, 3 (July 2001), 127–150. URL: <https://doi.org/10.1145/501786.501787>, doi: [10.1145/501786.501787](https://doi.org/10.1145/501786.501787). [2](#)
- [Mas21] MASTERMAP O. S.: Ordnance, 2021. URL: <https://www.ordnancesurvey.co.uk/business-government/products/mastermap-topography>. [11](#)
- [MZVWG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. *ACM SIGGRAPH* 26, 3 (2007), 85. [2](#)
- [PAAEZ20] PARK T., ALEXEI A., EFROS R. Z., ZHU J.-Y.: Contrastive learning for unpaired image-to-image translation. *ECCV* (2020). [4](#)
- [PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. In *ACM SIGGRAPH 2003 Papers*. 2003, pp. 313–318. [2](#)
- [PLWZ19] PARK T., LIU M.-Y., WANG T.-C., ZHU J.-Y.: Semantic image synthesis with spatially-adaptive normalization. In *IEEE CVPR* (2019), pp. 2337–2346. [1](#), [2](#), [3](#), [5](#), [6](#), [10](#)
- [PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proceedings of SIGGRAPH 2001* (2001), pp. 301–308. doi: [10.1145/383259.383292](https://doi.org/10.1145/383259.383292). [2](#)
- [RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241. [2](#), [4](#)
- [RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut" interactive foreground extraction using iterated graph cuts. *ACM SIGGRAPH* 23, 3 (2004), 309–314. [2](#)
- [Ser21] SERVICE D. O. S. W. M.: Digimap, 2021. URL: <https://digimap.edina.ac.uk/>. [2](#), [11](#)
- [TSK*19] TETERWAK P., SARNA A., KRISHNAN D., MASCHINOT A., BELANGER D., LIU C., FREEMAN W. T.: Boundless: Generative adversarial networks for image extension. In *IEEE ICCV* (2019), pp. 10521–10530. [4](#)
- [VAW*10] VANEGAS C. A., ALIAGA D. G., WONKA P., MÜLLER P., WADDELL P., WATSON B.: Modelling the appearance and behaviour of urban spaces. In *CGF* (2010), vol. 29, Wiley Online Library, pp. 25–42. [2](#)
- [VKW*12] VANEGAS C. A., KELLY T., WEBER B., HALATSCH J., ALIAGA D. G., MÜLLER P.: Procedural generation of parcels in urban modeling. In *CGF Eurographics* (2012), vol. 31, Wiley Online Library, pp. 681–690. [2](#)
- [WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (USA, 2000)*, SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., p. 479–488. URL: <https://doi.org/10.1145/344779.345009>, doi: [10.1145/344779.345009](https://doi.org/10.1145/344779.345009). [2](#)
- [WLKT09] WIE L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the Art in Example-based Texture Synthesis. In *CGF Eurographics* (2009), Pauly M., Greiner G., (Eds.), The Eurographics Association. doi: [10.2312/egst.20091063](https://doi.org/10.2312/egst.20091063). [2](#)
- [WLZ*18] WANG T.-C., LIU M.-Y., ZHU J.-Y., TAO A., KAUTZ J., CATANZARO B.: High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE CVPR* (2018), pp. 8798–8807. [2](#)
- [WZZH19] WU H., ZHENG S., ZHANG J., HUANG K.: Gp-gan: Towards realistic high-resolution image blending. In *Proceedings of the 27th ACM International Conference on Multimedia* (2019), pp. 2487–2495. [2](#)
- [YLY*19] YU J., LIN Z., YANG J., SHEN X., LU X., HUANG T. S.: Free-form image inpainting with gated convolution. In *IEEE ICCV* (2019), pp. 4471–4480. [7](#)
- [ZWS20] ZHANG L., WEN T., SHI J.: Deep image blending. In *The IEEE Winter Conference on Applications of Computer Vision* (2020), pp. 231–240. [2](#)
- [ZZB*18] ZHOU Y., ZHU Z., BAI X., LISCHINSKI D., COHEN-OR D., HUANG H.: Non-stationary texture synthesis by adversarial expansion. *ACM SIGGRAPH* 37, 4 (July 2018). URL: <https://doi.org/10.1145/3197517.3201285>, doi: [10.1145/3197517.3201285](https://doi.org/10.1145/3197517.3201285). [2](#)
- [ZZL19] ZHAN F., ZHU H., LU S.: Spatial fusion gan for image synthesis. In *IEEE CVPR* (2019), pp. 3653–3662. [2](#)
- [ZZP*17] ZHU J.-Y., ZHANG R., PATHAK D., DARRELL T., EFROS A. A., WANG O., SHECHTMAN E.: Toward multimodal image-to-image translation. In *NIPS* (2017). [4](#)