Autonomous Environment Texturing of Procedurally Generated Cities

OSCAR DADFAR, Carnegie Mellon University, USA
LINGDONG HUANG, Carnegie Mellon University, USA
HIZAL ÇELIK, Carnegie Mellon University, USA

We explore the possibility of producing photo-realistic and stylized videos from semantically segmented image sequences drawn from a procedurally generated interactive environment. By training style transfer algorithms such as Pix2Pix and Pix2PixHD on labeled street photographs from the Cityscapes dataset, we can apply these stylizations onto our pre-recorded segmented city to autonomously texture our environments. We further evaluate our pipeline using the GTA V image dataset to showcase its feasibility on large interactive scenes for 3D animation and game texturing. Our algorithm can be used to support the texturing pipeline by lessening the need for human-design and intervention when creating 3D interactive environments.

CCS Concepts: • **Computing methodologies** → **Texturing**; **Procedural animation**; *Non-photorealistic rendering*.

Additional Key Words and Phrases: style-transfer, autonomous texturing, computer games.

ACM Reference Format:
Oscar Dadfar, Lingdong Huang, and Hizal Çelik. 2021. Autonomous Environment Texturing of Procedurally Generated Cities. 1, 1 (April 2021), 6 pages. https://doi.org/10.1145/1122445.1122456

Authors' addresses: Oscar Dadfar, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA; Lingdong Huang, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA; Hizal Çelik, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA.

1 INTRODUCTION

Texturing 3D environments is a laborious process requiring hundreds to even thousands of artist hours when giving color and texture to large environments. The production of large scale video games such as GTA V had more than 49 map artists and 37 environment artists working full time to design and texture the fictional city of San Andreas [8]. The more recent Red Dead Redemption 2 hired more than 178 full time artists to take players back into its its wild west culture [9]. Other titles such as No Man's Sky [7] tried to avoid the high cost of hand-made environment development by procedurally generating planets for a unique gameplay experience, but still relied on artist-crafted designs and textures to populate these worlds with color and detail.

To help with the development of 3D scenes for video games and animations, we produced a procedurally generated 3D city autonomously textured using style transfer to give both life and color to our environment without the need for human environment modeling or texturing. Our work is the first of its kind to completely autonomize both the modeling and rendering pipeline of 3D environments, providing a unique interactive environment experience to users every time.

2 RELATED WORKS

Early style-transfer models would extract features from a stylized image and apply these features onto a target image, minimizing

Fig. 1. Results of our city block-splitting algorithm.

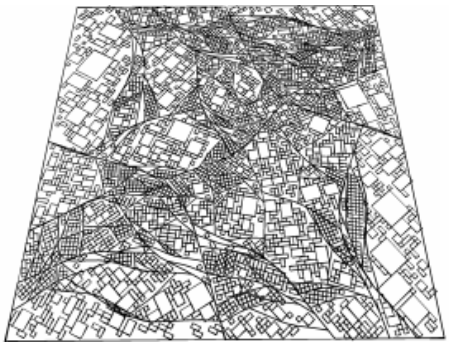

Fig. 2. Results of our building-planning algorithm.

the stylistic-loss from the style image and content loss from the target image [3] [11] [22]. Such models can be used to create [13] and argument [21] stylized scenes from real-life images, leading to a new wave of creative imagery. Style-transfer is both sensitive to adjustments in the target image [17] while also being computationally expensive [5], leading to the exploration of temporally consistent style-transfer among video frames. Such work predicts an optical flow trajectory of elements in the scene [16] by reparameterizing their content and style loss across adjacent frames in video [18] [20]. When applying style transfer to our environment, we aim to maintain temporal consistency in both lighting and content.

Many style-transfer algorithms can be used to help create rich semantic segmentations from real-life scenes [10] [19], providing vision applications with a color-coded vocabulary of the world [4]. Such tools can also be used the other way around to generate real-life scenes from segmentations [12] and to help augment semantic details to create new images [2]. We aim to create a semantic segmented 3D environment given that drawing semantic segmentations is simpler than drawing real-life images [14]. This way, we can use style transfer to go from our minimally-colored environment to full realistic textures.

Style-transfer research has been used before in 3D game development. Google's Stadia can use real-time shaders with stylistic parameters sent from a video style-transfer algorithm to redraw video frames in different styles [6]. These parameter weights are pre-computed and can be subbed in with different stylistic parameters at runtime, allowing developers to filter between a collection of artistic styles in-game. Their approach learns features from a single image, and recolors the entire target video game frame equally. We instead aim to texture frames based on the semantics of the objects in view, providing a more comprehensive texturing mechanism where buildings would get separate texturings from trees and so forth.

3 METHOD

Our work is comprised of two parts: the first part involves procedurally generating a unique color-coded representation of a 3D interactive city, while the second part involves pre-training a GAN on a semantic segmented city dataset and applying the style-transfer to a recorded image sequence from interacting in the environment.

3.1 Procedural City Generation

3.1.1 Splitting Blocks. We generate the roads first from a top-down perspective in Figure 1. We start by considering the whole map as a single large polygon and recursively subdivide polygons by connecting lines between two points on the edge of a previous polygon. We make a cut passing through the mid-points of the two longest borders of the largest polygon, helping to minimize the the variance in polygon sizes. To reproduce the curved shape of roads in real life, we apply a perlin noise filter to offset points along a straight line.

3.1.2 Planting Buildings. We further divide city blocks into buildings in Figure 2. A grid is laid upon the block of interest with a random rotation, where all squares on the grid that lays outside the block are deleted. Remaining squares are randomly combined to make larger rectangles, providing variance in building size. The maximum and minimum size of the buildings depends on both the area of the block and the granularity of the grid cells.

We can extrude the shapes of the roads and buildings to create a 3D representation of our city in Figure 3. The buildings near the center of the map are made skinnier and taller to represent downtown, and buildings near edge of the map are made larger and flatter to represent suburban areas. Roads generated by the first couple recursive iterations are drawn wider to represent main streets, while roads generated by the last few recursions are drawn narrower to represent local roads.

3.1.3 Populating the City. Simple miscellaneous structures such as street lights, traffic lights and street signs in Figure 4 are generated using groups of cubes with some variation on the height and thickness. We place these structures adjacent to the roads.

To generate trees, we write a procedural blob generator which applies some noise to a triangulated sphere. Each vertex on the sphere is randomly offset from the center, with the magnitude generated from perlin noise. We stitch together several of these blobs to a lengthened cube to create trees. The same blob method is also used to generate an abstract mesh for humans, where we use one blob for the head and one blob for the body. We add cars and busses as pre-modeled assets.

3.1.4 Animation. We animate vehicles across roads and humans adjacent to roads. Humans are allowed to cross intersections when

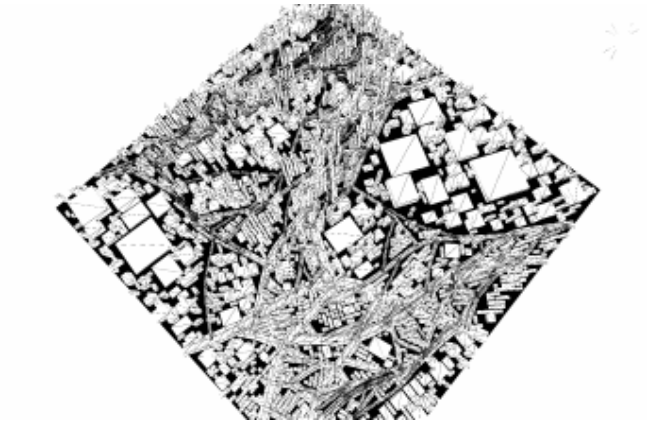

Fig. 3. Results of our extrusion algorithm.

multiple roads meet. Entities spawn in a random location and are given a target location across the map. Upon reaching their destination, a new target location is selected. At each time step, the entity can either move forward or backward along the road, or turn into another road if it is at a crossroad. The decision is informed by its desire to reach the destination. If two entities bump into each other, one of them will be stunned for a couple seconds, allowing the other to pass. To simplify intersections, all traffic keeps to the right. Intersections alternate between allowing vehicles to pass and allowing pedestrians to cross.

We permit the user to move and fly around the city, allowing the user to capture interesting overhead shots of buildings and vehicles. The user is also allowed to take control and drive around vehicles as if in an actual video game.

3.1.5 Coloring. Each object is given a unique solid coloring depending on the class of object it is. An example coloring is given in Figure 5. We match the segmentation colorings to that of the dataset trained on. Shadows and edges are removed from objects so as not to interfere with the style transfer, as it uses the color-coded semantics to determine the object class and texturing required.

3.2 Style Transfer

3.2.1 Datasets. We used the Cityscapes dataset [1] for training our Pix2Pix [10] model. When training on the Cityscapes dataset

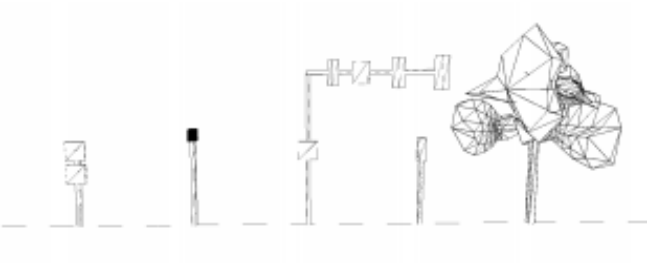

Fig. 4. Miscellaneous structure examples.

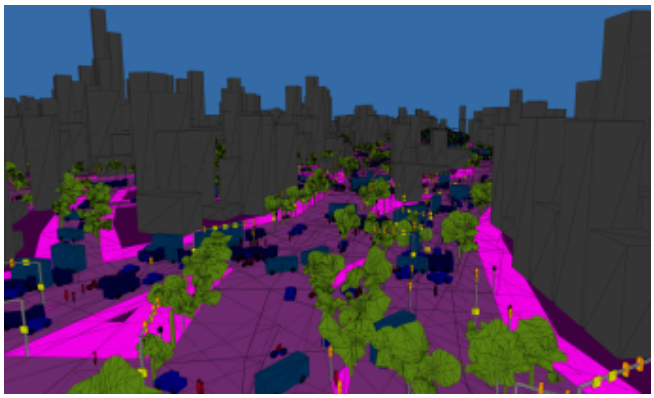

Fig. 5. Generated city scene populated with vehicles and people.

for approximately 40 epochs, we fond the most saturated image yielded far too fake of an image. The original training set is based on footage from various German cities, resulting in a rather dim, downcast image set that was too gray and monotone for enjoyable results. To fix this, we trained on a color corrected version of the Cityscapes dataset that ended up increasing the brightness and contrast of the results, providing clearer texturing details.

We trained additional models using Pix2Pix and Pix2PixHD [19] on the GTA V dataset [15] while aiming for a more creative and stylized video-game look with these models. We trained another instance of Pix2PixHD on only a tenth of the GTA V images for more epochs. We also extracted the night-time images from the GTA V dataset and trained separately on those, producing a nighttime texture mode for our city.

3.2.2 Testing. We capture frames of a user interacting in our procedurally generated segmented city and ran the sequence of frames through our Pix2Pix model to style-transfer the results. The input image sequence from our environment is a set of RGB frame segmentations. The colors are compressed by the video encoder capturing the frames, leading to color offsets in some pixels. To fix this, we store the most popular colors and their corresponding ID's in a data structure, binning stray colors to their closest popular color to recalibrate outlier pixels.

Because our generated video was individual Pix2Pix frames strung together into a video, slight inconsistencies between the frames seemed to cause an unwanted flickering effect in the video. We resolved this by linearly interpolation between consecutive frames and then inserting those interpolated frames between the original frames to increase framerate and smoothness while decreasing flickering.

4 RESULTS

,

We trained two separate Pix2Pix models: one on the original 2048x1024-pixel Cityscapes dataset for 80 epochs, and one on a downsampled 512x256-pixel Cityscapes dataset for 160 epochs using GTX 980 Ti's. We further trained separate Pix2Pix and Pix2PixHD models on the GTA V dataset for 200 and 120 epochs respectively.

Fig. 6. Examples of the Pix2Pix style-transfer trained on downsampled color-corrected images of the Cityscapes dataset.

Fig. 7. Comparing the pretrained Pix2Pix Cityscapes model to a model trained on the raw dataset and the recolorized downsampled dataset. The recolored downsampled dataset is able to train on more epochs while providing higher-contrast and clearer results.

Fig. 8. Comparing Pix2Pix against Pix2PixHD. Models were trained on the full 25,000 images, while a smaller Pix2PixHD model was trained on 2,500 images for more epochs. The Pix2Pix model suffered from rapid color flickering while the Pix2PixHD model was able to learn a more consistent lighting representation.

Fig. 9. Setting the height of vehicles above the ground to test the robustness of our style transfer algorithm on unseen vehicle views.

To test the texturing capabilities, we had users interact with their own procedurally generated color-coded city to collect an image sequence of segmentations. We ran these seqences through our style transfer algorithm to generate automatic texturings for each user's environment.

5 DISCUSSION

We are able to achieve populated, diverse scenes with each procedural generation of our map. The results of our city animations is an amusing primitive form of traffic system. While many traffic accidents are effectively prevented and where vehicles and people generally get to go where they intend to go, traffic jams and serial car accidents occur frequently at busy crossroads. Sometimes they run into a Klotski puzzle situation, and the road becomes so blocked that nobody can pass through, accumulating more and more stuck cars over time. A potential solution could be to use a third-party AI-assisted driving mechanism provided by most game engines to help resolve collisions and stalls in vehicle and human interactions.

The Pix2Pix results on the Cityscapes dataset in Figure 7 produced blurry and dim outputs for the pretrained (weights originally provided by Pix2Pix) and 2048x1024-pixel models. We predict this is because the original images were too cloudy with low contrast. To

Fig. 10. Skyline view of procedural city on the GTA V dataset.

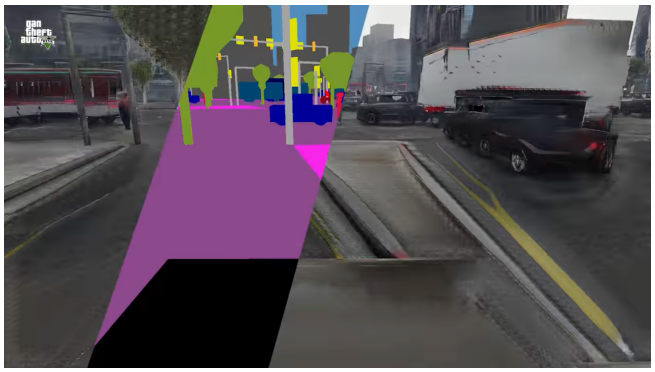

Fig. 11. Comparison of the procedural city before and after the autonomously texturing.

improve performance, we downsampled our images and increased brightness and contrast, allowing for more training on sharper and more exposed images. Our results produced higher contrast images with sharper details as we were able to train for more epochs on the downsampled dataset.

Several results of the Cityscapes dataset in Figure 6 look collage-like, given that we are trying to map photo-realistic textures to low-poly geometry. The GTA V dataset provides a more cartoon-like look that matches the more simplistic geometry from our city, giving it more of a video game feel.

When style-transferring on the GTA V dataset in Figure 8, our Pix2Pix results would flicker a lot between frames, most likely due to the GTA V dataset being comprised of both day and night images. Pix2PixHD on the other hand did not have this issue due to the model training with temporal consistency in mind. At earlier epochs of Pix2PixHD, the results would have too high contrast as the model would clash day and night textures. As we increase the number of epochs, the results become more cohesive with an overall warmer global lighting as the model learns the average lighting features in the dataset.

Training on only a tenth of the images for ten times as many epochs in Figure 8 led to blurrier images most likely due to the limited dataset blending image textures together from different scene lightnings. This signified the need for larger datasets on diversely lit scenes. The Pix2PixHD model trained on only the night-time dataset provided a consistently dark environment. This is ideal for scenes that want to simulate day and night environments. A sunset environment with sunset images could potentially be added to bridge the two together to simulate a full day of stylizations.

The Pix2Pix model could process frames at roughly 7 frames a second on a GTX 980 Ti, but the image transfer overhead from interacting with a city on a local device and sending the frames over to the compute server to style-transfer made any real-time performance difficult. Developers could instead attempt to host the game and style transfer over the server and stream frames back to the user's device for faster frame rates, especially as cloud gaming becomes more popular. The Pix2PixHD model took several seconds

per frame, so potential developers could choose to use Pix2Pix in-game and Pix2PixHD for cutscenes and animations.

6 CONCLUSION

Our approach generated "dream-like" and "artistic" renderings of a completely procedurally generated city without any human intervention when generating and texturing the scene. Our findings can be used to create 3D video game and animation worlds autonomously rather than having artists pre-texture everything. There are also a growing number of datasets to train with. Similar to the GTA V dataset, individuals can intercept the video card to retrieve frames and other meta-data needed to easily segment data from different video games, adding to the library of video-game inspired textures. As these technologies advance, the generated imagery will become more realistic as well, allowing animators and game designers a chance at creating unique procedural environments every time.

REFERENCES

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. *CoRR* abs/1604.01685 (2016). arXiv:1604.01685 http://arxiv.org/abs/1604.01685

[2] Aysegul Dundar, Ming-Yu Liu, Ting-Chun Wang, John Zedlewski, and Jan Kautz. 2018. Domain Stylization: A Strong, Simple Baseline for Synthetic to Real Image Domain Adaptation. *CoRR* abs/1807.09384 (2018). arXiv:1807.09384 http://arxiv.org/abs/1807.09384

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2015. A Neural Algorithm of Artistic Style. *CoRR* abs/1508.06576 (2015). arXiv:1508.06576 http://arxiv.org/abs/1508.06576

[4] Emanuele Ghelfi, Paolo Galeone, Michele De Simoni, and Federico Di Mattia. 2019. Adversarial Pixel-Level Generation of Semantic Images. *CoRR* abs/1906.12195 (2019). arXiv:1906.12195 http://arxiv.org/abs/1906.12195

[5] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. 2017. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *CoRR* abs/1705.06830 (2017). arXiv:1705.06830 http://arxiv.org/abs/1705.06830

[6] Google. 2019. *Behind the Scenes with Stadia's Style Transfer ML.* https://stadia.dev/blog/behind-the-scenes-with-stadias-style-transfer-ml/

[7] HelloGames. 2016. No Mans Sky. [CD-ROM].

[8] IGDB. 2013. *Grand Theft Auto V Credits.* https://www.igdb.com/games/grand-theft-auto-v/credits

[9] IGDB. 2019. *Red Dead Redemption 2.* https://www.igdb.com/games/red-dead-redemption-2/credits

[10] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR* abs/1611.07004 (2016). arXiv:1611.07004 http://arxiv.org/abs/1611.07004

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*.

[12] Peilun Li, Xiaodan Liang, Daoyuan Jia, and Eric P. Xing. 2018. Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption. *CoRR* abs/1801.01726 (2018). arXiv:1801.01726 http://arxiv.org/abs/1801.01726

[13] Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. 2020. Diverse Image Generation via Self-Conditioned GANs. (2020). arXiv:2006.10728 [cs.CV]

[14] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic Image Synthesis with Spatially-Adaptive Normalization. *CoRR* abs/1903.07291 (2019). arXiv:1903.07291 http://arxiv.org/abs/1903.07291

[15] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. 2016. Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision (ECCV) (LNCS, Vol. 9906)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, 102–118.

[16] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2016. Artistic style transfer for videos. *CoRR* abs/1604.08610 (2016). arXiv:1604.08610 http://arxiv.org/abs/1604.08610

[17] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* abs/1607.08022 (2016). arXiv:1607.08022 http://arxiv.org/abs/1607.08022

[18] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. Video-to-Video Synthesis. *CoRR* abs/1808.06601 (2018). arXiv:1808.06601 http://arxiv.org/abs/1808.06601

[19] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2017. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *CoRR* abs/1711.11585 (2017). arXiv:1711.11585 http://arxiv.org/abs/1711.11585

[20] Xide Xia, Tianfan Xue, Wei sheng Lai, Zheng Sun, Abby Chang, Brian Kulis, and Jiawen Chen. 2020. Real-time Localized Photorealistic Video Style Transfer. (2020). arXiv:2010.10056 [cs.CV]

[21] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. 2020. Differentiable Augmentation for Data-Efficient GAN Training. arXiv:2006.10738 [cs.CV]

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR* abs/1703.10593 (2017). arXiv:1703.10593 http://arxiv.org/abs/1703.10593