GAN Theft Auto: Autonomous Texturing of Procedurally Generated Interactive Cities

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

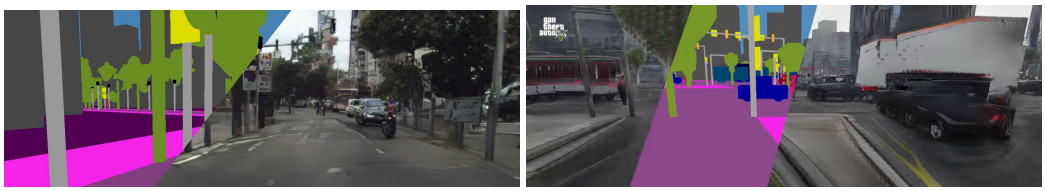

Fig. 1. Cross-views of Cityscape [top] and GTA V [bottom] texturings using Pix2PixHD.

We explore the possibility of producing photo-realistic and stylized videos from semantically segmented image sequences drawn from a procedurally generated interactive 3D environment. We evaluate our environment using the Cityscape and ACDC weather dataset to obtain swappable daytime, nighttime, and various weather texturings from our city. We further use the GTA V image dataset to showcase its feasibility on large interactive scenes for 3D animation and game texturing, demonstrating the ability to repurpose existing video game textures when generating our city. Our algorithm can be used to support video games by autonomizing both the environment generation process, as well as supporting researchers by providing a semantic testing environment for many city style-transfer algorithms. Video documentation at: https://tinyurl.com/vb63k87x

CCS Concepts: • **Applied computing** → **Media arts**; • **Computing methodologies** → *Rasterization.*

Additional Key Words and Phrases: Style-Transfer, Autonomous Texturing, Computer Games.

ACM Reference Format:
Oscar Dadfar, Lingdong Huang, and Hizal Çelik. 2021. GAN Theft Auto: Autonomous Texturing of Procedurally Generated Interactive Cities. 1, 1 (October 2021), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Texturing 3D environments is a laborious process requiring thousands of artist hours when creating 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 [9], and the more recent Red Dead Redemption 2 hired more than 178 full time artists to create its wild west culture [10]. Other titles such as No Man's Sky [8] tried to avoid the high cost of hand-made environment development by

Authors' addresses: Oscar Dadfar, Carnegie Mellon University, Pittsburgh, USA, odadfar@andrew.cmu.edu; Lingdong Huang, Carnegie Mellon University, Pittsburgh, USA, lingdonh@andrew.cmu.edu; Hizal Çelik, Carnegie Mellon University, Pittsburgh, USA, hizalc@andrew.cmu.edu.

XXXX-XXXX/2021/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

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 style transfered 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 create a rich interactive 3D scene modeled after the segmentation data used in training city style-transfer algorithms. We demonstrate its effects on various style-transfer programs and datasets to providing a unique interactive environment experience to users every time without the need for any human intervention.

2 RELATED WORKS

Procedurally generating city geometry can be achieved using perlin or fractal noise [13], with the goal of achieving some form of tiling structure similar to a voronoi diagram. City generation can also be modeled after the L-systems form of biological development [16] [26] or after a 2D overhead image input[22] [2]. We consider an input-less generation strategy, allowing us to procedurally generate a new city each time for the user. We defer texturing to using style transfer as we focus on building a city with semantic colorings only.

Early style-transfer models would extract features from a stylized image and apply these features onto a target image, minimizing the stylistic-loss from the style image and content loss from the target image [4, 12, 29]. Such models can be used to create [15] and argument [28] 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 [23] while also being computationally expensive [6], leading to the exploration of temporally consistent style-transfer among video frames by predicting the optical flow trajectory of elements in the scene [20, 24, 27]. 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 [11, 25], providing vision applications with a color-coded vocabulary of the world [5]. Such tools can also be used the other way around to generate real-life scenes from segmentations [14] and to help augment semantic details to create new images [3]. We aim to create a semantic segmented 3D environment given that drawing semantic segmentations is simpler than drawing real-life images [17]. 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. Nvidia's generative driving environment [24] can be used to recreate existing video footage as a segmented 3D city that can further be style-transferred and moved around in, but it generates the same scene and texturing each time without interactive elements. We aim to extend their results by providing a algorithm for procedurally generating unique segmented 3D cities with interactive elements such as driving cars and walking pedestrians with swappable texturings. The more recent [18] leverages semantic segmentations in their model, but focuses on style transferring the existing city in GTA rather than making a novel environment to interact in. Google's Stadia uses real-time shaders with stylistic parameters sent from a video style-transfer algorithm to redraw video frames in different styles [7]. 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.

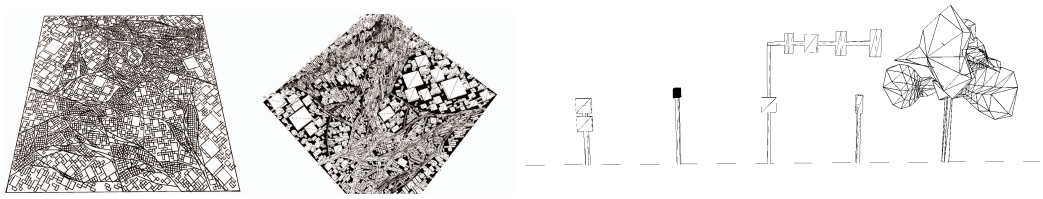

Fig. 2. Results of our [Left] building-planning algorithm with [Right] extruding.

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 applying a style-transfer model of a target dataset to image sequences from interacting in the environment.

3.1 Procedural City Generation

3.1.1 Splitting Blocks. We consider 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 divide city blocks into buildings on the left of 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 the middle of Figure 2. 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 on the right of Figure 2 are generated using groups of cubes with some variation on the height and thickness. We place these structures adjacent to the roads.

Most trees in the segmentation datasets are illustrated as blob-like structures where the leaves are coalesced into one blob shape. To be consistent with the segmentations, we write a procedural blob generator which applies some noise to a triangulated sphere. Each vertex on the sphere is randomly offset using perlin noise from the center. We stitch together several of these blobs 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 generate cars and busses as a few rectangles and spheres stitched together.

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

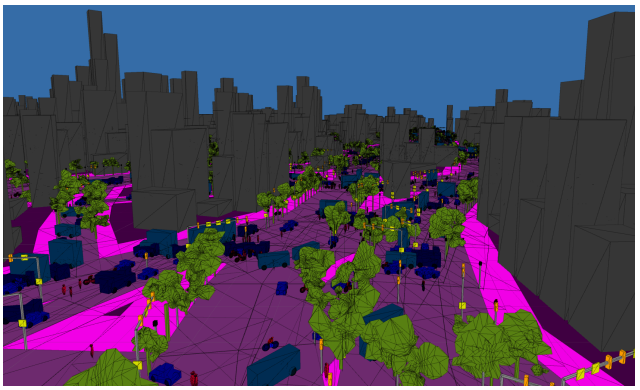

Fig. 3. Generated city wireframe scene populated with interactive vehicles and people that users can walk and drive around.

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 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 3. 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 Model. We evaluate our environment on several GAN-based style-transfer methods, including Pix2Pix, Pix2PixHD in Figure 4 and Vid2Vid in Figure 5. We configured these models to use a UNet generator and a convolutional discriminator with 4 downsample layers and 9 residual blocks with out generator outputting frames of size 1024x512. Modifying the number of generator filters varied the model complexity the most, so we used an evolutionary search strategy to identify the best number of generator filters to maximize both inference speeds and quality for each dataset. We grid-search over several configs of generator filters from 16 to 128. After each 50 epochs, we ask 3 participants to score the visual quality of the models on a validation set. We subtract from this model score a value proportional to the number of parameters of the model as a means of regularizing the scores to promote models with lower complexity. We continue training the upper half of scored models using an evolutionary search strategy. Table 1 lists the ideal configs for each model and dataset that maximizes on inference runtimes and quality.

3.2.2 Datasets. We used the Cityscapes dataset [1] of 5,000 images for training. When training for approximately 40 epochs, we found 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

Fig. 4. Comparing Pix2Pix against Pix2PixHD. Models were trained with 64 generator features 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. 5. Vid2Vid trained with 128 generator features on Cityscapes dataset. Frames are temporally consistent, minimizing flickering while also learning a more cohesive global illumination of the scene.

contrast of the results, providing clearer texturing details. We further trained on the ACDC city weather dataset [21] with around 500 images per weather condition, providing us with 4 distinct weather configurations, including snow, rain, fog, and night. We trained additional models on the GTA V dataset [19] comprised of 25,000 images, aiming for a more creative and stylized video-game look with these models. We trained additional instances on only a tenth of the GTA V images for more epochs to experiment with a smaller model. We also extracted the night-time images from the GTA V dataset and trained separately on those, producing a nighttime video game texture mode.

3.2.3 Testing. The input image sequence from our environment is a set of RGB frame segmentations such as in Figure 3. The colors are compressed by the video encoder, 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. We can capture the instance map from our generated city and use that during the evaluation stage for models such as Vid2Vid. By loading in multiple model weights at startup, we can switch between different dataset weights on demand, providing us with a way of shifting between different weathering looks and styles on the fly.

Table 1. Performance Benchmarks

Model				Config	
Model	**Dataset**	**Size**	**Epochs**	**GenFilters**	**EvalFPS**
Pix2Pix	Cityscape	5K	80	48	24.6
Pix2Pix	Cityscape (clr)	5K	260	48	24.6
Pix2Pix	GTA V	25K	200	64	18.5
Pix2PixHD	Cityscape	5K	200	48	21.4
Pix2PixHD	GTA V	25K	120	64	15.3
Pix2PixHD	GTA V (tiny)	2.5K	200	48	21.4
Pix2PixHD	GTA V (dark)	600	200	40	24.8
Pix2PixHD	ACDC (night)	506	200	32	31.5
Pix2PixHD	ACDC (rain)	500	200	32	31.5
Pix2PixHD	ACDC (fog)	500	200	32	31.5
Pix2PixHD	ACDC (snow)	485	200	32	31.5
Vid2Vid	Cityscape	5K	200	128	6.7

[a]EvalFPS processed on a GTX 3060 Ti. clr = re-colored.

4 RESULTS

We trained several models on varying datasets in Table 1 using GTX 980 Ti's. To test the texturing capabilities, we had users interact with their own unique 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. We evaluated both the procedurally-generated city and our style transfer algorithm on a GTX 3060 Ti in order to identify FPS on a consumer-level device, reporting the rate at which each pretrained model could output stylized frames in Table 1. We include additional video documentation here: https://tinyurl.com/vb63k87x

Fig. 6. Pix2PixHD trained on the ACDC Weather dataset for 200 epochs. Each dataset contained roughly 500 images.

5 DISCUSSION

We are able to achieve populated, diverse scenes with each procedural generation of our map in Figure 1 where no two cities are the same due to the random noise we use. 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 sometimes occur at busy crossroads. In these cases, 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 in vehicle and human interactions.

The ACDC Weather results in Figure 6 suffer from a collage-like look, 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. This, coupled with how the GTA V dataset had 5x more images than Cityscapes and 50x more images than ACDC made it more robust to various views provided by our city.

Given the processing speeds of the style transfer algorithms in Table 1, consumer-grade GPUs could handle real-time support for most of these texturings. When optimizing the generator size for inference speeds, we found a correlation between the dataset size and the generator complexity needed to accurately retain styling capabilities. Smaller datasets such as ACDC did not require as complex a model for inference and could run at a higher FPS than Cityscape. GTA being the largest of the datasets required the highest model complexity to retain its higher-quality results, leading it to run the slowest. The Vid2Vid Cityscape model performed at a substantially lower framerate compared to the other models primarily because it computed the optical flow between images, adding substantially to the feature complexity and creating frame dependencies in the inferencing scheme. For more demanding inference models such as Vid2Vid, developers could instead attempt to host the game and style transfer over the server with higher-throughput devices and stream frames back to the user's device for faster frame rates, especially as cloud gaming such as Google Stadia [7] become more popular.

6 CONCLUSION

Our approach is able to generate artistic renderings of a completely procedurally generated city without any human intervention when generating and texturing the scene. We demonstrate that our semantic city can be used by research scientists to help evaluate their city-based style transfer algorithms on novel scenes, and by game developers looking to create automatically generated and textured environments. Users can also swap between different styles at will, finding a style that is right for them. 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] Rozenn Dahyot, G. Lacey, K. Dawson-Howe, François Pitié, and D. Moloney. 2016. PatchCity: Procedural City Generation using Texture Synthesis.

[3] 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

[4] 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

[5] 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

[6] 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

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

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

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

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

[11] 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

[12] 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*.

[13] George Kelly and Hugh McCabe. 2006. A survey of procedural techniques for city generation. *Institute of Technology Blanchardstown Journal* 14 (01 2006).

[14] 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

[15] 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]

[16] Yoav I. H. Parish and Pascal Müller. 2001. Procedural Modeling of Cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 301–308. https://doi.org/10.1145/383259.383292

[17] 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

[18] Stephan R. Richter, Hassan Abu AlHaija, and Vladlen Koltun. 2021. Enhancing Photorealism Enhancement. arXiv:2105.04619 [cs.CV]

[19] 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.

[20] 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

[21] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. 2021. ACDC: The Adverse Conditions Dataset with Correspondences for Semantic Driving Scene Understanding. *ArXiv e-prints* (2021). arXiv:2104.13395 [cs.CV]

[22] Bingyu Shen, Boyang Li, and Walter J. Scheirer. 2021. Automatic Virtual 3D City Generation for Synthetic Data Collection. In *2021 IEEE Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 161–170. https://doi.org/10.1109/WACVW52041.2021.00022

[23] 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

[24] 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

[25] 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

[26] Xiao Wang, Yacheng Song, and Peng Tang. 2020. Generative urban design using shape grammar and block morphological analysis. *Frontiers of Architectural Research* 9, 4 (2020), 914–924. https://doi.org/10.1016/j.foar.2020.09.001

[27] 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]

[28] 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]

[29] 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