SURFGenerator: Generative Adversarial Network Modeling for Synthetic Flooding Video Generation

Stephen Lamczyk
The Department of Electrical and Computer Engineering
Vision Lab
Old Dominion University
Norfolk, Virginia
slamc001@odu.edu

Kwame Ampofo
The Department of Electrical and Computer Engineering
Vision Lab
Old Dominion University
Norfolk, Virginia
kampo001@odu.edu

Behrouz Salashour
The Department of Civil and Environmental Engineering
Old Dominion University
Norfolk, Virginia
bsala001@odu.edu

Mecit Cetin
The Department of Civil and Environmental Engineering
Old Dominion University
Norfolk, Virginia
mcetin@odu.edu

Khan M. Iftekharuddin
The Department of Electrical and Computer Engineering
Vision Lab
Old Dominion University
Norfolk, Virginia
kiftekha@odu.edu

***Abstract*—The current lack of labeled large-scale flooding video datasets hinders work in semantic video segmentation of flooding images. Flooding image segmentation is needed in many critical application areas, including recurrent flooding analysis due to sea-level rise and climate change. To address the lack of labeled flooding videos, this work proposes a novel generative adversarial network (GAN)-based Synthetic Urban Recurrent Flooding (SURF) Generator (SURFGenerator) for generating synthetic videos of flooding at different water depth levels. We first generate large-scale training samples of synthetically rendered videos of flooding information using physics-based water simulation tools within Blender and Mantaflow for our model. A two-part model is then developed, inspired by previous work, composed of a masker network using LinkNet followed by an off-the-shelf vid2vid painter network. The masker inference network is trained with rendered images and corresponding binary segmentation masks to identify potential flooding areas for a specific depth level in the video. Finally, the painter network is trained separately using segmented render videos as input. The painter network takes the masks and initial video and generates synthetic water in the masked areas to create an output flooding video. To the best of our knowledge, for the first time in literature, this work generates synthetic flooding videos with physically relevant features such as moving ripples and small waves when cars move on the flooded streets. These synthetically generated realistic flooding videos may be used to generate large-scale labeled images for semantic segmentation and water depth estimation on flooded streets and other urban settings using deep network models.**

***Index Terms*—Synthetic Video Generation, Flooding, Dataset, Generative Adversarial Networks**

I. Introduction

Among all-natural disasters, flooding frequently occurs around the world, causing significant damage. Flash floods have caused 1075 fatalities recorded from 1996 to 2014 in the United States [1]. The prevalence of weather-related disasters is increasing rapidly, and flood risk management has become an important challenge for many nations [2]. Image data from various sources (e.g., satellite, cell phones, drones, surveillance cameras) has become a valuable source for flood risk and damage assessment [2]. Researchers have been developing computer vision and deep learning methods to detect floodwater on roadways [3], [4] and estimate water depth [5]. Compared to common pressure or ultrasonic sensor-based systems, camera-based methods using computer vision are considerably less expensive [6]. Although many works have studied the problem on a larger scale by using satellite imagery and digital elevation models (DEM) [6], [7], there are a few studies using local street view flood images. Most of these methods use a region of interest (ROI) over a time interval and detect changes using a form of a differencing technique [8]. For example, Yu and Hahn [9] have developed a remote detection and monitoring system using narrow band channel to detect changes in the water level of a river indicated on a bridge pole. However, their model might not be robust with variations in the scene as it was trained only for one location.

The frequency of nuisance flooding is increasing around the world and along the eastern coast of the United States of America. Nuisance flooding brings significant inconvenience for motorists, especially if there is no information available about their route during such occurrences. A key challenge with image-based methods for flood monitoring methods is the lack of large-scale labeled datasets representing complex scenes including variations in lighting, shadows, ripples in floodwater, etc. The framework presented in this work is

This material is based upon work supported by the National Science Foundation under Grant No. 1951745

expected to aid a comprehensive real-time flood monitoring system that can identify floods from traffic surveillance videos, predict the extent of the flood (flood depth/level), alert motorists with this information, and reroute motorists based on this information to safer routes.

In a flood detection and warning system, detection of floodwater depth is naturally required. For example, a motorist may decide it is safe to drive through 2 cm of floodwater but not 20 cm. However, there is a distinct lack of diversity in preexisting flood datasets [5] that are composed of different flood levels. In our experience, training a model on these datasets does not translate well into the real world. In addition, these datasets are composed of independent images that have no temporal relationship to each other as opposed to our rendered video data where the frames are related to each other. In a real-time flood detection system, the input would realistically be a live video feed from traffic cameras.

Computationally differentiating a water body in an image can be a very challenging task, and many filtering methods have been used to determine its segmentation [6]. Edge detectors and comparing histograms of images in grayscale or color have been studied when dealing with more stable sceneries [6]. Data-driven methods based on Machine learning (ML) have been used to overcome these shortcomings and improve performance [10]. The use of ML for flood prediction and water detection has increased immensely in recent years, and models with multiple research strategies (hybrid models) are the most used. However, most of these methods still need heavy preprocessing of the data which can vary from one location to another [10]. Deep learning methods can help by transforming the data at one level from the raw input and have the potential to work more robustly in a diverse scenery [11]. However, in most cases very large datasets are needed to avoid overfitting, which is not the case in many realistic settings [12]. To address these challenges, we propose a GAN-based Synthetic Urban Recurrent Flooding (SURF) Generator (SURFGenerator) to create synthetic video images of flooding on streets to support research on a real-time flood management system which can warn motorists about floods and the extent of floods. The proposed SURFGenerator is novel since it can generate flood scenes for a video (as opposed to still images), captures complex fluid dynamics (e.g., ripples and waves) due to moving vehicles on partially inundated roads, and generate varying depths of floodwater.

The remainder of this paper is as follows. Section II discusses the appropriate background for understanding the tools used to generate the synthetic flooding data. Recent advances and the history of this field are also discussed as well as the architecture of the models used in this pipeline. Section III describes SURFGenerator for synthetic video generation and the training and testing process for the models. Section IV discusses the results from SURFGenerator, and Section V concludes and discusses our future work in this field.

II. Background

A. Related Work

Previous related work, namely [13], [14], [15], and [16], focused mainly on synthetic flooding image generation and experimented with various networks and pipelines to achieve this. The latest work in this series [13] utilizes a masker network composed of subnetworks to generate a singular mask where flooding is likely to be in an input image. The painter network then takes the mask and input image and paints in flooding where the mask indicates flooding is likely. However, as stated previously, the goal of this paper is to create synthetic flooding videos that could potentially be used in other domains such as semantic segmentation and depth estimation. Ideally, the semantic segmentation and depth estimation would be performed in real-time on live video streams from traffic cameras. Therefore, the methods used in the literature will not be ideal for our use case. Since these models are trained on still images, they would not generate realistic synthetic videos. This issue exacerbates if cars or other objects were moving through the water.

In addition, the method used to generate synthetic flooding images in these works is inherently flawed. The Unity3D game engine used in some of these works inherently limits the realism of potential rendered flooding images as all frames must be rendered in real-time. Other software can be utilized to generate more realistic images at the expense of rendering time. In addition, the water body simulation in Unity3D consists of just a flat plane superimposed on top of a scene. This is not true to life or physically correct.

B. Blender and Mantaflow

Blender [17] is a very powerful and open-source 3D creation suite capable of rendering videos and simulating physically correct water through its inclusion of Mantaflow [18] - a fluid simulation framework developed by the Technical University of Munich. This fluid simulation framework can be used in the presence of other 3D models to create realistic scenes of flooding. In our case, we use this fluid simulation framework with a 3D model of a city as this generates pertinent training data for a semantic segmentation model that needs to detect flooding within cities. We re-utilized the masks that were generated for segmentation and instead use them to train the masker and painter.

C. Masker

LinkNet[19] is a fully convolutional neural network that, in our experience, works well with performing semantic segmentation of images of flooding. Given an input image $x \in \mathbb{R}^{mxnx3}$, LinkNet, with a sigmoid activation function and only one output channel, outputs a segmentation mask $y \in \{0,1\}^{mxnx1}$ with 1 representing flooding and 0 representing everything else. We describe in the methods section how we used LinkNet as a component in our masker network to generate flooding segmentation masks for large-scale input videos. We found that LinkNet performs better over different model architectures to perform semantic segmentation of potential inundation.

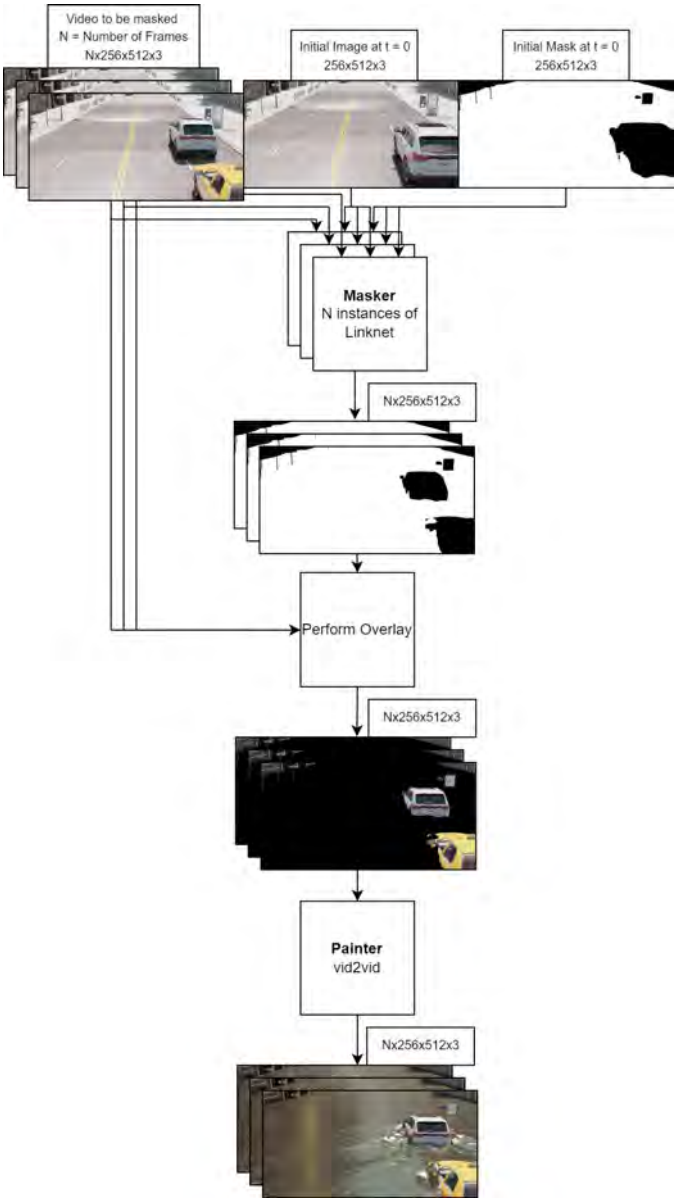


Fig. 1. Overall Pipeline

D. Painter

The painter consequently takes the masks from the masker and the raw video frames to generate the output video. We utilized vid2vid [20] for our painter portion. While vid2vid was initially created to regenerate videos from their output semantic segmentation video masks, we found that it worked well for our intended purpose of generating synthetic flooding in a masked area.

III. Methods

A. Synthetic Image Generation

The first step in creating realistic videos of flooding is to obtain a lifelike environment. We used a 3D model based loosely around New York City [21]. After obtaining the model, several important details were changed to make the environment look more realistic.

Fig. 2. Example picture rendered in Blender with no water present.

The first step in creating a realistic model was to add lighting effects to the city street lights and cars as they were originally nonfunctional. Both of these are key for accurate reflections in night-time images of flooding. In addition, we had to tune shaders to create accurate reflections as the car paint was originally matte. This is obviously not realistic, so we added reflections and glossiness to both.

Once these problems were fixed, car movement was then added using key frames. The car 3D models were also a singular mesh, so the car wheels did not rotate when the car did. This resulted in inaccurate water simulations when the car moved through the water. So, we separated the car wheel meshes from the car models and tied the car wheel rotation to the car's linear movement.

The next step in creating realistic images of flooding was to add the water to the scene. We created a water simulation domain that included the city street. The water is then added to the domain and allowed to settle before the cars move through. The amount of water can be varied to create training data for depth estimation. A compromise was struck between the water quality and simulation performance. Accurate but fast foam generation was a big problem. The default parameters for foam generation were set too high and resulted in overwhelming amounts of foam. Final parameters for water simulation are shown in Table I. Using these parameters results in a water simulation that takes about 3-4 days to run using an Intel(R) Xeon(R) Gold 6130 @ 2.1GHz.

After establishing the framework for the water simulation, we rendered data for both the masker and painter. All renders were done using the Cycles rendering engine on four NVIDIA V100 16GB GPUS. The NVIDIA OptiX denoiser was used, and the noise threshold was set to .01. Six light bounces were used. These settings result in a rendering time of about two minutes per frame.

1) Painter Training Data: After setting up the scene, we focused on rendering the synthetic data to train the painter. We rendered both dry and flooded videos. We rendered the three flooded conditions in Table II from eight different perspectives. These eight different perspectives include differences in camera angle, camera height, and lens type. We rendered 300 sequential video frames for each condition and perspective

TABLE I
SETTINGS USED IN BLENDER TO SIMULATE WATER

Setting	***Value***
Resolution Divisions	512
Time Scale	0.5
CFL Number	4
Use Adaptive Time Steps	True
Time Steps Maximum	4
Time Steps Minimum	2
Simulation Method	APIC[22]
System Maximum	0
Particle Radius	1
Sampling	2
Randomness	0.1
Particles Maximum	16
Particles Minimum	8
Narrow Band Width	3
Fractional Obstacles	False
Viscosity	False
Diffusion	True
Foam	True
Spray	False
Bubbles	False
Combined Export	False
Upres Factor	1
Wave Crest Potential Maximum	160
Wave Crest Potential Minimum	40
Trapped Air Potential Maximum	400
Trapped Air Potential Minimum	100
Kinetic Energy Potential Maximum	100
Kinetic Energy Potential Minimum	20
Potential Radius	2
Potential Update Radius	2
Wave Crest Particle Sampling	200
Trapped Air Particle Sampling	40
Particle Life Maximum	25
Particle Life Minimum	10
Particle in Boundary	Delete
Mesh	True
Upres Factor	2
Particle Radius	2
Use Speed Vectors	False
Mesh Generator	Final
Smoothing Positive	1
Smoothing Negative	1
Concavity Upper	3.5
Concavity Lower	0.4

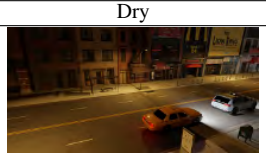
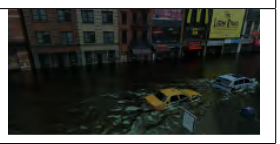
combination. This results in 7,200 flooded images or 24 flooded videos.

The next step is to create masks using Blender. This is trivial to do automatically. Each mesh must be given a render layer pass index corresponding to its semantic segmentation label. In this case, the labels are 1 for flooded and 0 for not flooded.

After generating the masks, we then use them to mask off the frames to create our training data for the painter. This dataset will be referred to as X_P, Y_P for the images and segmentation masks respectively. The painter training data is suited to training the painter to paint any initially masked off video, real or synthetic, as it represents many combinations of perspective, and lighting.

2) Masker Training Data: The dry images are rendered based off the three dry conditions in Table II and from eight different perspectives as in the generation of the training data

TABLE II
SIX CONDITIONS RENDERED

	Dry	Dirty Water
Dark		
Dim		
Bright		

for the painter module. For the generation of the masker masks, we generated 10 different flooding masks for different depth levels outlined by [5] in Table III. This is accomplished by setting the water level within Blender according to the 10 different depth levels and then generating the mask for the image as stated above. This means that for each dry image, there are ten different potential masks for flooding that are used in training, so there are 7,200 input images and 72,000 output masks. To improve temporal consistency between frames, instead of a one-to-one mapping between frames and the mask, we also input the initial frame at t = 0 and the initial mask at t = 0 into the network for training. It is important to note that this means the masker is not entirely unsupervised, as an initial mask is still needed to initialize the video generation. Furthermore, this was made under the assumption that the camera will remain stationary throughout the duration of the video as this was the case with our data. However, the masker can automatically perform alterations to the first mask in case cars or other objects pass through the water. Similarly to the painter, all images and masks are 256x512 resolution. This dataset will be referred to as X_M, Y_M for the images and segmentation masks respectively.

TABLE III
TEN DIFFERENT FLOODING MASK DEPTH LEVELS GENERATED [5]

Level	Depth Level (cm)
Level 1	1
Level 2	10
Level 3	21
Level 4	43
Level 5	64
Level 6	85
Level 7	106
Level 8	128
Level 9	149
Level 10	170

B. Masker

Consider a masker network with a LinkNet [19] decoder with an EfficientNetB0 [23] encoder **M**. We utilized EfficientNetB0 instead of the original LinkNet encoder. The goal for the masker is to take a series of dry input images $X = [x_0, x_1, ..., x_T]$ and a reference mask y_0 for frame 0 and generate a series of masks $Y = [y_1, ..., y_T]$ that indicate where flooding is likely to be for each frame in this video. Moreover, since there are ten different depth levels of flooding $L = \{l_1, l_2, ..., l_{10}\}$, the masker must learn to generate Y each of these at any specific different depth level $l_i \in L$ given a reference mask y_0 that represents a mask of potential inundation at the given level l_i. This reference mask y_0 along with the corresponding initial frame x_0 are used at each timestep to give the model information about the desired flood level and to maintain improved temporal stability over the video.

The masker **M**, is trained using the mask training dataset X_M, Y_M. For each initially unflooded video $X \in X_M$, $X = [x_0, x_1, ..., x_T]$ where each frame $x_i \in \mathbb{R}^{256x512x3}$. For each mask representing the potential flooded pixels over a video $Y \in Y_M$, $Y = [y_0, y_1, ..., y_T]$ where each frame of the mask video $y_i \in \{0,1\}^{256x512x1}$. Given input X and Y_0, the output Y is calculated as follows. Each frame $x_i \; in \; X$, is fed to a instance of **M** so that each output segmentation mask $y_i = \mathbf{M}(x_0, y_0, x_i)$. Again, we mention that the reference values of y_0 and x_0 are fed to **M** for each time step to give information about the target depth level of the flood.

We then train the masker network on the sequences of training data while withholding a 15% validation split. This means that 1,080 frames of the original 7,200 are used for validation. During training and testing we randomly shuffle the data within these splits. Limited by 16GB of video memory, we train the network with a batch size of 4 utilizing Adam [24] optimizer and Dice loss [25] on y_t. We found that Dice loss performed better over different loss functions such as binary focal loss and binary cross-entropy. The dice coefficient is given as the following formula and is a quantity we want to maximize. For the binary segmentation case we are utilizing here, TP represents true positives, FP represents false positives, and FN represents false negatives:

$$DiceScore = \frac{2TP}{2TP + FP + FN}$$

C. Painter

After training the masker, we train the vid2vid[20] network P to perform the new task of filling in water in the image where the masker believes water should be for a specific depth level. We use the ground truth from X_P, Y_P instead of the masker's predictions for the training. During inference, ideally the masker's predictions can be inputted, but these can also be overridden with manually created masks. We create the training data for the masker as follows. Given a series of frames $X \in X_P$, $X = [x_0, x_1, ..., x_T]$ where each frame is again $x_i \in \mathbb{R}^{256x512x3}$ a series of binary masks $Y = [y_0, y_1, ..., y_T]$ is generated by the masker **M**. The painter input W can be formulated as $W = [x_0 \odot \bar{y_0}, x_1 \odot \bar{y_1}, ..., x_T \odot \bar{y_T}]$ where $\odot$ represents the Hadamard Product. The painter **P** then calculates the painter output Z as $\mathbf{P}(W) = Z$ where Z is an output RGB video. This means that both W and Z are $\mathbb{R}^{T+1x256x512x3}$

During training, we utilize the default training parameters for the vid2vid network and firstly train the model on 128x256 resolution frames and then 256x512 resolution frames. We also partition our inputs and outputs to be 30 frames long. We keep the rest of the model parameters the same.

IV. Results and Discussion

For the inference step, a series of frames is first fed through the masker. We use the masker output masks in the painter network to obtain the final synthetic flooding videos. Our masker converged to .113 validation loss with 90 % accuracy. Detailed final statistics are shown in Table IV. Dice loss curves, indicating mismatch in the overlap of the labeled and computer masks, are shown in Figure 3. Figure 4 shows training and validation accuracy for the masker over time. Due to a lack of a benchmark dataset, it is difficult to quantitatively compare our results with those of other works. Figure 5 shows sample qualitative results compared with results from [13]. Visual inspection of images generated using SURFGenerator suggests the results are noticeably better. The models proposed in the literature [13] appear to struggle to find the appropriate flooded areas. While our work does obscure the cars close to the camera with the mask, it still adjusts the mask in the background based on cars driving through the street. This is most obvious in the top right corner of the frames. We also provide x_0 and y_0 for our model under the reference row in 5 indicated by Ref. Since no reference is required for ClimateGAN, it is not shown and is N/A.

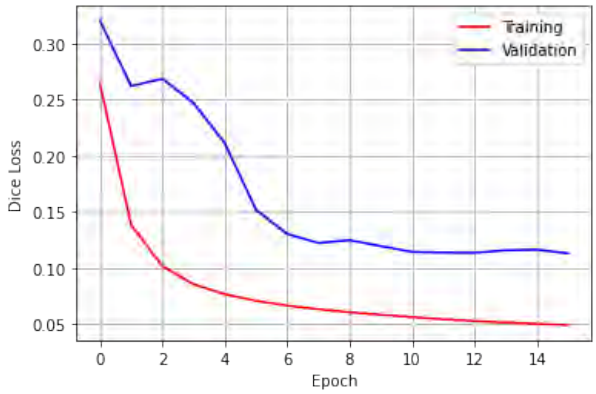


Fig. 3. Training and Validation Dice Loss Curves

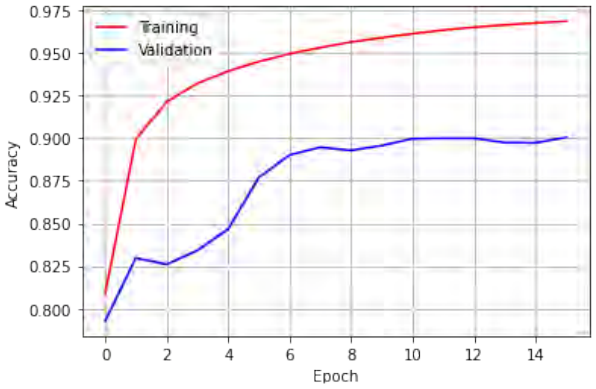


Fig. 4. Training and Validation Accuracy Over Time

TABLE IV
FINAL QUANTITATIVE RESULTS FROM MASKER

Statistic	Train	Validation
Dice Loss	.05	.113
Binary Accuracy	.97	.90
mIOU	.93	.83
Precision	.94	.90
Recall	.98	.90
F1 Score	.95	.89
F2 Score	.96	.89

We further attach with this paper four separate video file examples that show the proposed novelty of this work in rendering fine details such as water ripples and small waves as a car moves on the flooded streets in comparison to other works.

- Painter Results
  - SURFGenerator: https://youtu.be/lirlI7IbI1E
  - ClimateGAN: https://youtu.be/kUoELXDZOzg
- Painter Results
  - SURFGenerator: https://youtu.be/k6DQPmUaJKA
  - ClimateGAN: https://youtu.be/hPjOmadLet8

These four videos utilize the same hand-labeled masks to ensure a fair comparison of the painter modules. It is evident that there is some interaction between the synthetic water and the cars driving through the water in our results. This is completely lacking in previous works [13] as they only generate synthetic still images. The painter module still has the same problem as the masker where there is no established benchmark dataset to quantitatively compare the results.

V. CONCLUSION AND FUTURE WORK

This work implements a novel GAN-based SURFGenerator for generating synthetic videos of flooding at different water depth levels. We utilized LinkNet as our masker network and train the masker to output a mask using flooding example images. We trained the vid2vid network to utilize the masks from LinkNet to generate synthetic water on an image using the mask as a reference. The results show that SURFGenerator is an effective method of generating synthetic flooding videos

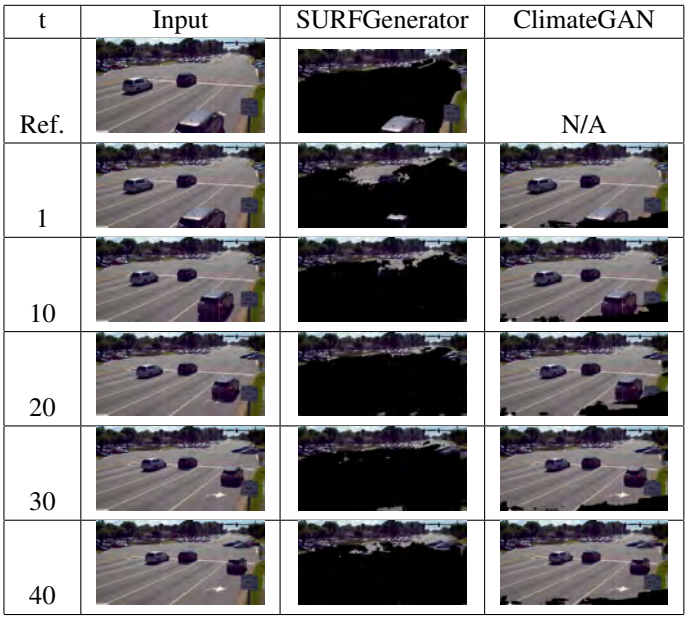


Fig. 5. Masks Compared With [13]. Reference mask is not given for ClimateGAN as their work does not need one.

in contrast to previous work that only generated synthetic images. We are also currently generating a hand-labeled real flooded video dataset that can be used as a benchmark for future work. One limitation of SURFGenerator is low-resolution output due to video memory limitations. In the future, we plan to improve SURFGenerator to generate higher quality output. SURFGenerator may also be used to generate synthetic videos of flooding at different depths to train a water depth level estimation network. In addition, we plan to improve our current semantic segmentation model that detects floodwaters and the depth of flooding in images.

VI. ACKNOWLEDGEMENTS

This research has been partially supported by National Science Foundation grants 1951745 and 1828593 at Old Dominion University. We would also like to thank Alexander Glandon of the Old Dominion University Vision Lab for proofreading this work.

REFERENCES

[1] G. Terti, I. Ruin, S. Anquetin, et al. "A Situation-Based Analysis of Flash Flood Fatalities in the United States". In: *Bulletin of the American Meteorological Society* 98.2 (2017), pp. 333–345. DOI: 10.1175/BAMS-D-15-00276.1. URL: https://journals.ametsoc.org/view/journals/bams/98/2/bams-d-15-00276.1.xml.

[2] N. S. Romali, Z. Yusop, M. Sulaiman, et al. "FLOOD RISK ASSESSMENT: A REVIEW OF FLOOD DAMAGE ESTIMATION MODEL FOR MALAYSIA". In: *Jurnal Teknologi* 80.3 (Feb. 2018). DOI: 10.11113/jt.v80.11189. URL: https://journals.utm.my/jurnalteknologi/article/view/11189.

[3] M. Witherow, C. Sazara, I. Winter-Arboleda, et al. “Floodwater detection on roadways from crowdsourced images”. In: *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 7 (June 2018), pp. 1–12. DOI: 10.1080/21681163.2018.1488223.

[4] C. Sazara, M. Cetin, and K. M. Iftekharuddin. “Detecting floodwater on roadways from image data with handcrafted features and deep transfer learning¡sup¿*¡/sup¿”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 804–809. DOI: 10.1109/ITSC.2019.8917368.

[5] P. Chaudhary, S. D’Aronco, J. Leitão, et al. “Water level prediction from social media images with a multi-task ranking approach”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 167 (2020), pp. 252–262. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2020.07.003. URL: https://www.sciencedirect.com/science/article/pii/S0924271620301842.

[6] B. Arshad, R. Ogie, J. Barthelemy, et al. “Computer Vision and IoT-Based Sensors in Flood Monitoring and Mapping: A Systematic Review”. In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: 10.3390/s19225012. URL: https://www.mdpi.com/1424-8220/19/22/5012.

[7] L. Hawker, P. Bates, J. Neal, et al. “Perspectives on Digital Elevation Model (DEM) Simulation for Flood Modeling in the Absence of a High-Accuracy Open Access Global DEM”. In: *Frontiers in Earth Science* 6 (2018). ISSN: 2296-6463. DOI: 10.3389/feart.2018.00233. URL: https://www.frontiersin.org/article/10.3389/feart.2018.00233.

[8] B. Arshad, R. Ogie, J. Barthelemy, et al. “Computer Vision and IoT-Based Sensors in Flood Monitoring and Mapping: A Systematic Review”. In: *Sensors* 19.22 (2019). ISSN: 1424-8220. DOI: 10.3390/s19225012. URL: https://www.mdpi.com/1424-8220/19/22/5012.

[9] K. Kim, N.-K. Lee, Y. Han, et al. “Remote Detection and Monitoring of a Water Level Using Narrow Band Channel”. In: *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*. ISPRA’07. Corfu Island, Greece: World Scientific, Engineering Academy, and Society (WSEAS), 2007, pp. 25–30. ISBN: 1237890123456.

[10] A. Mosavi, P. Ozturk, and K.-w. Chau. “Flood Prediction Using Machine Learning Models: Literature Review”. In: *Water* 10.11 (2018). ISSN: 2073-4441. DOI: 10.3390/w10111536. URL: https://www.mdpi.com/2073-4441/10/11/1536.

[11] R. Bentivoglio, E. Isufi, S. N. Jonkman, et al. “Deep Learning Methods for Flood Mapping: A Review of Existing Applications and Future Research Directions”. In: *Hydrology and Earth System Sciences Discussions* 2021 (2021), pp. 1–43. DOI: 10.5194/hess-2021-614. URL: https://hess.copernicus.org/preprints/hess-2021-614/.

[12] A. Antoniou, A. Storkey, and H. Edwards. *Data Augmentation Generative Adversarial Networks*. 2018.

[13] V. Schmidt, A. S. Luccioni, M. Teng, et al. *ClimateGAN: Raising Climate Change Awareness by Generating Images of Floods*. 2021. arXiv: 2110.02871 [cs.CV].

[14] G. Cosne, A. Juraver, M. Teng, et al. *Using Simulated Data to Generate Images of Climate Change*. 2020. arXiv: 2001.09531 [cs.CV].

[15] A. Luccioni, V. Schmidt, V. Vardanyan, et al. “Using Artificial Intelligence to Visualize the Impacts of Climate Change”. In: *IEEE Computer Graphics and Applications* 41 (Jan. 2021), pp. 8–14. DOI: 10.1109/MCG.2020.3025425.

[16] V. Schmidt, A. Luccioni, S. K. Mukkavilli, et al. “Visualizing the Consequences of Climate Change Using Cycle-Consistent Adversarial Networks”. In: *ArXiv* abs/1905.03709 (2019).

[17] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[18] N. Thuerey and T. Pfaff. *MantaFlow*. *http://mantaflow.com*. 2018.

[19] A. Chaurasia and E. Culurciello. “LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation”. In: *CoRR* abs/1707.03718 (2017). arXiv: 1707.03718. URL: http://arxiv.org/abs/1707.03718.

[20] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, et al. “Video-to-Video Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[21] 3DE. *3D Model City Modular New - TurboSquid 1165557*. June 2017. URL: https://www.turbosquid.com/3d-models/3d-model-city-modular-new-1165557.

[22] C. Jiang, C. Schroeder, A. Selle, et al. “The Affine Particle-in-Cell Method”. In: *ACM Trans. Graph.* 34.4 (July 2015). ISSN: 0730-0301. DOI: 10.1145/2766996. URL: https://doi.org/10.1145/2766996.

[23] M. Tan and Q. V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[24] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[25] F. Milletari, N. Navab, and S.-A. Ahmadi. “V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation”. In: *2016 Fourth International Conference on 3D Vision (3DV)*. 2016, pp. 565–571. DOI: 10.1109/3DV.2016.79.