**DyNCA:** Real-Time Dynamic Texture Synthesis Using Neural Cellular Automata

Ehsan Pajouheshgar*, Yitao Xu*, Tong Zhang, Sabine Susstrunk

**Problem**

Textures are everywhere. We perceive them as spatially repetitive patterns. Dynamic Textures are textures that change over time inducing a sense of motion.

**Cellular Automata and Conway’s Game of Life**

- **Grid:** State of the cell at location $i, j$ at time $t$ is $S_{ij}^t$.
- **Neighborhood:** Cells can perceive their neighbors.
- **Update rule:** How cell states change at each step.

**Game of Life**

- Survive if 2 or 3 neighbors are alive
- Become alive if 3 neighbors are alive
- Die or remain dead

**Cellular Automata can be implemented using Convolutions, Gilpin [2019].**

**Architecture**

- **Cell States:** State is a $c$ dimensional vector. The first 3 dimensions are the RGB values.
- **Perception:** Four fixed convolution kernels that are frozen during the model’s training.
- **Multi-scale Perception:** Increase the communication range of the cells and improve stability.
- **Positional Encoding:** Allows the cells to be aware of their global position in the grid.
- **Positional Encoding:** The update rule is represented by two trainable FC layers and a random binary update mask.

**Contributions**

- (I) Arbitrary Resolution
- (II) Arbitrary long videos
- (III) Synthesize new samples
- (IV) Real-time video editing
- (V) Require pretrained models
- (VI) Disentangled appearance and motion.
- (VII) Vector field supervision.

**Comparison and User Study**

- We show videos to the participants and ask them to choose the video that appears the most realistic.
- (A) Tesfaldet et al. [2018]. (B), and (C) two different configurations from Xie et al. [2017].

**Results**

- Top: DyNCA captures the appearance and the motion from a video.
- Bottom: DyNCA also disentangles the appearance and motion and performs Dynamic Style Transfer when the target dynamics are different.

**Training and Loss Functions**

- **Synthetic Frame:** $S^f = \frac{1}{3} \sum_{i,j \in \text{mask}} (K_{fb}(S_{ij}) + K_{fg}(S_{ij}))$
- **Optical Flow:** $\mathcal{L}_{optical} = \left\| \mathcal{L}(S_{ij}, P_{ij}) + \mathcal{L}(P_{ij}, S_{ij}) \right\|_1$
- **Style Matching Loss:** $\mathcal{L}_{style} = \sum_{i,j \in \text{mask}} \left( \frac{1}{c} \sum_{k=1}^{c} \left( \frac{1}{h} \sum_{i,j \in \text{mask}} \left( \frac{1}{w} \sum_{k=1}^{w} \left( \frac{1}{c} \sum_{l=1}^{c} (S_{ij,k}^{l} - mean_{i,j,k}^{l})(P_{ij,k}^{l} - mean_{i,j,k}^{l}) \right)^2 \right) \right)^2 \right)$

**Link to our Demo**

dynca.github.io