Quantum Collapse Algortihm

Kerlin Michel

1 Introduction

The Wave Function Collapse algorithm[1], a texture synthesis algorithm, was made by Gumin and is heavily based on Merrell's Model Synthesis thesis paper[2]. In a high level description the models synthesis algorithm creates an randomly generate output, image or 3d model, that locally looks similar to some input. The algorithm interprets the input and output as being locally partitioned into parts. When the algorithm finishes succesfully each output part will be equal to some input part. The model synthesis initializes all output parts to be in state where each input part is a possible assignment. The algorithm then loops through each output part and randomly assign it to an input part and then possible output assignments for unassigned output parts are updated based on an adjaceny contraint. The adjaency constaint is built by analyzing the adjaceny of parts in the input. The Wave Function Collapse works similarly with the following differences:

1. Instead of iterating through the output in a scanline manner like in Model Synthesis the output is iterated through by choosing the output with the lowest entropy, the output with least number of possible input parts that can be assigned to it

2. Wave Function Collapse doesn't implement a feature of Model Synthesis where the output is modified in sections rather than as a whole.

Merrell examines the difference between Model Synthesis and Wave Function Collapse in Comparing Model Synthesis and Wave Function Collapse[3].

This project is meant to examine and generalize both algorithms. Implementing the algorithm using quantum computing will also be explored.

List of Algorithms

2 Quantum Collapse Algorithhm

The Quantum Collapse Algorithhm randomly generates an output that is composed of parts that are local similar to some input. The algorithm will assign a

part in the input to a part in the output. The input parts are described by the set $S = \{s_1, \ldots, s_{k-1}\}$ where k is the number of possible states, $k = |S|$.

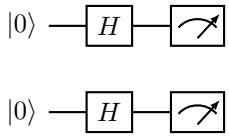
The output O is represented by a $d_0 \times \cdots \times d_{D-1} \times k$ array where D is the number of dimensions of the output and $d_i \in \mathbb{Z}^+$. For images $D = 2$ and for videos and 3D models $D = 3$. Each output part is described by $o_{d_0,\ldots,d_{D-1}}$ where is an array: $o_{d_0,\ldots,d_{D-1}} = [O_{d_0,\ldots,d_{D-1},0},\ \ldots,\ O_{d_0,\ldots,d_{D-1},k-1}]$. Each entry in a o represents the possible states of an output part where each index of corresponds to a state in S. If an entry in some o is > 0 then the the state in S that corresponds to the index of said entry is a possible assignment to the output part. The algorithm initializes each entrpy O to 1 so that all output parts, $o_{d_0,\ldots,d_{D-1}}$, can be assigned any state in S.

When a state s is assign to some o all entries in o become 0 except for the entry at the index that corresponds to the state s being assigned. When a o is assigned other o's states will be updated for which states are possible to assign to them based on a neighborhood constraint. The neighborhood constraint restricts which states can be assigned near each other and is created by examining the neighborhood of the input parts. To construct a neighborhood constraint the neighborhood needs to be defined. The neighborhood, N, is $\{(v_0^0, \ldots, v_{D-1}^0),\ \ldots,\ (v_1^{|N|-1}, \ldots, v_{D-1}^{|N|-1})\}$ where $v_j^i \in \mathbb{Z}$. Each v^i in N specifies that some $o_{d_0,\ldots,d_{D-1}}$ is a neighbor to some other $o_{d_0+v_0^i,\ldots,d_{D-1}+v_{|N|-1}^i}$. v^i is essentially an index offset. The neighborhood constraint, C_n, is then represented by a $k \times |N| \times k$ array. Every entry $c_{i,j,k}$ in C_n, where indexes i and k correspond to two states and j corresponds to some neighborhood offset is either equal to 1 if this neighborhood constraint is in the input and 0 if not.

3 Quantum Computing

3.1 Quantum Randomness

One application of quantum computing is the ability to produce truly random numbers. For example the following quantum circuit will randomly generate a 2 bit number where 00, 01, 10, 11 are all equally likely:



References

1. GUMIN, M., 2016. Wave function collapse, https://github.com/mxgmn/wavefunctioncollapse.

2. MERRELL, P. 2009. Model Synthesis. PhD thesis, University of North Carolina at Chapel Hill.

Algorithm 1 Quantum Collapse Algorithhm

$S \leftarrow \{s_1, s_2, \ldots, s_k\}$
$N \leftarrow \{(v_0^0, \ldots, v_{D-1}^0),\ \ldots,\ (v_1^{|N|-1}, \ldots, v_{D-1}^{|N|-1})\}$
$O \leftarrow [d_0] \ldots [d_{D-1}][k]$
$P \leftarrow$ CreateStack()
$C_n \leftarrow [k][|N|][k]$

for all entries $O[d^0] \ldots [d^{D-1}][s]$ in O **do**
    $\mathrm{O}[d^0] \ldots [d^{D-1}][s] \leftarrow 1.0$ ▷ Set output parts to a superposition of all states
end for

while $\nexists o_{d_0,\ldots,d_{D-1}} \neq [0, \ldots, 1, \ldots, 0]$ **do** ▷ Loop until ouput fully collapses
    $d_0, \ldots, d_{D-1} \leftarrow$ SelectUncollapsedOutputPart(O)
    $\mathrm{O}[d^0] \ldots [d^{D-1}] \leftarrow$ Collapse($d_0, \ldots, d_{D-1}$)
    push(P, $(d_0, \ldots, d_{D-1})$)
    while P is not empty **do**
        $v \leftarrow$ pop(P)
        for $n \in N$ **do**
            shouldPropagate $\leftarrow$ false
            for s = 0 to $k - 1$ **do**
                if $\mathrm{O}[(v + n)][s] > 0$ **then**
                    if $C_n[\mathrm{O}[v][s] = 1][\mathrm{n}][\mathrm{s}] = 0$ **then**
                        $\mathrm{O}[(v + n)][s] \leftarrow 0$
                        shouldPropagate $\leftarrow$ true
                    end if
                end if
            end for
            if shouldPropagate **then**
                push(P, $(d_0 + v_0,\ \ldots,\ d_{D-1} + v_{D-1})$)
            end if
        end for
    end while
end while

3. MERRELL, P. 2021. Comparing Model Synthesis and Wave Function Collapse, https://paulmerrell.org/wp-content/uploads/2021/07/comparison.pdf