**Legorizer**

**Introduction**

I implement python program that take voxelized file (.vox) and calculate the Lego schematic for it. I implement the Legorizer in Python by use **Legorization from silhouette-fitted voxelization1** as a guide

**The code**

The code can be clone from github with instruction

<https://github.com/flapperz/legorizer>

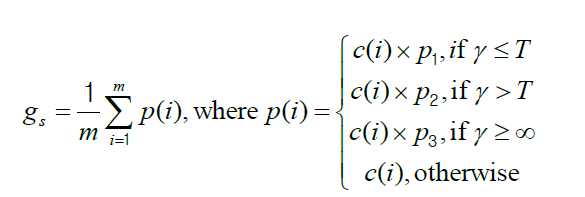
You can limit kind of brick in legorized model by edit `config.py`

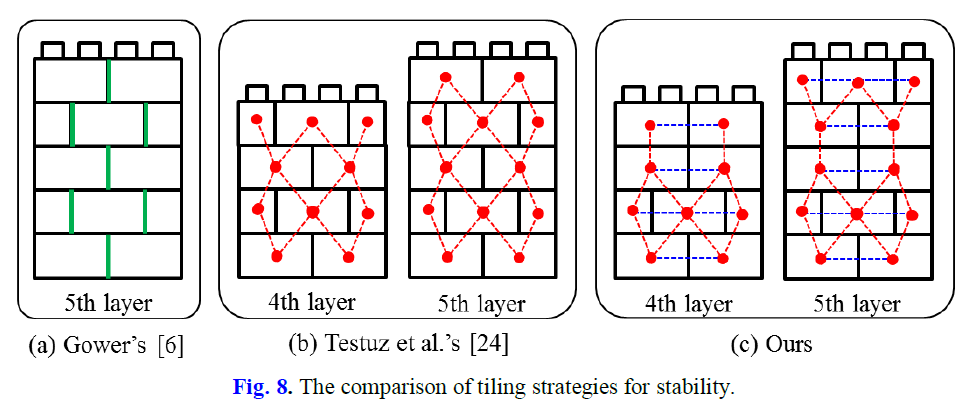
**Explanation**

First I implement .vox file read/write function the output accompany the use of Graph coloring to distinguish each brick.

Then, I adopt the the A\* search algorithm with stability heuristic from the papers

The stability term <GS> as follow





To use stability term as the paper suggest I have also implement 2-color graph structure to store Lego brick as nodes

The paper also propose another 2 heuristic: Aesthetic and Efficiency which I didn’t implement in this Legorizer. Unfortunately, the stability heuristic alone seem to be unbalance for A\* result in for ever search for optimum answer which is not feasible for my machine so I add the term

GF = - tiling level

This term encourage the A\* to progress to another level

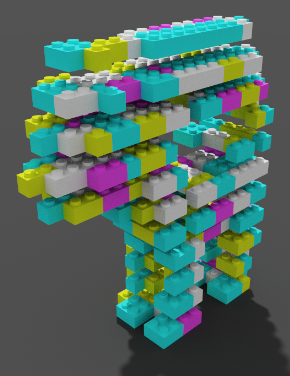
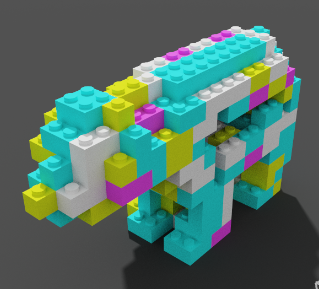
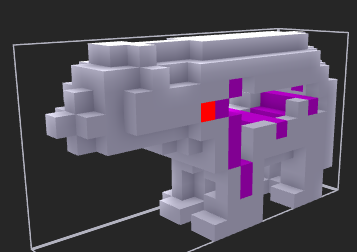
**Result**

****

**Model:** Bear 6x20x21

**Brick Set:** [(1,1), (1,2), (1,3), (2,2), (2,4), (2,6), (2,3), (2,8)]

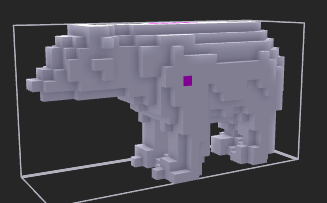
**Execution Time:** 19.301 s

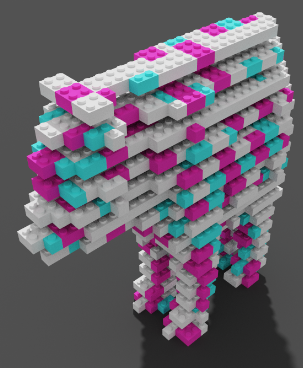
****

**Model:** Bear 6x20x21

**Brick Set:** [(1,1), (1,2), (1,3), (2,2), (2,4), (2,6), (2,3), (2,8)]

**Execution Time:** 1149 s

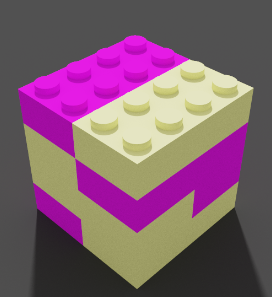
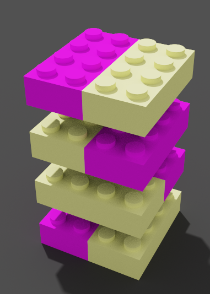
****

****

**Model:** 4x4x4 cube

**Brick Set:** [(1,1), (1,2), (1,3), (2,2), (2,4), (2,6), (2,3), (2,8)]

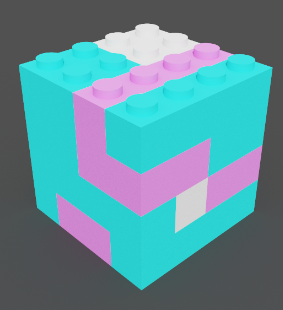
**Execution Time:** 0.1847 s

** **

**Model:** 4x4x4 cube

**Brick Set:** [(1,4), (2,2)]

**Execution Time:** 0.4123 s

**** 