Additional Materials

In Figure 2, 3, and 4, we present additional analysis for the shape registration and geometric optimization methods. In Figure 5, we show a pattern design using templates with edge color constraints. Finally, In Figure 6 and Figure 7, we show analyses of the Gurobi solver's performance in terms of scalability and intermediate solutions.

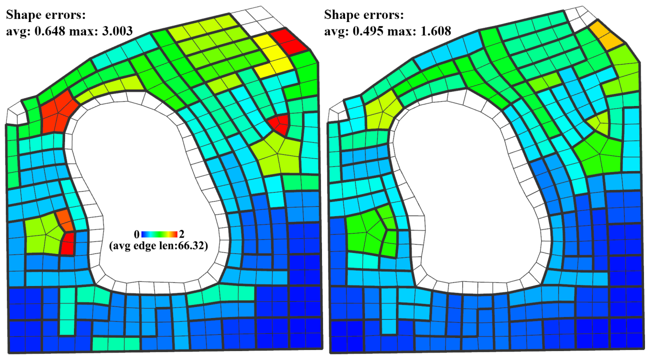

Figure 2: *Left and right: before and after a geometric optimization that minimizes the shape registration errors of tiles for tiling Figure 10 (f).*

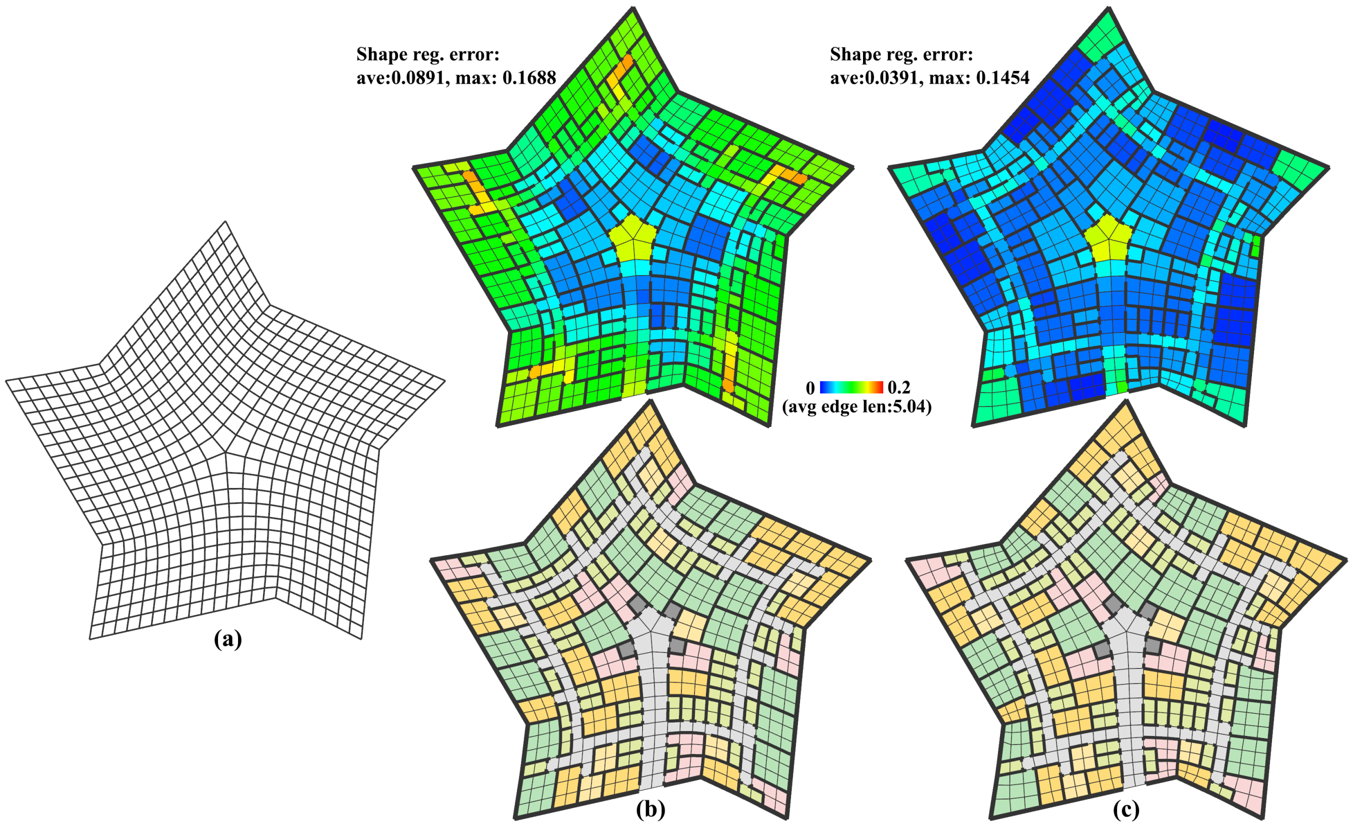

Figure 3: *Geometric optimization for the pentagon-shaped building example. (a) The quadrangulated problem domain. (b) The tiling for the ground floor before geometric optimizations. (b) The mesh after geometric optimization.*

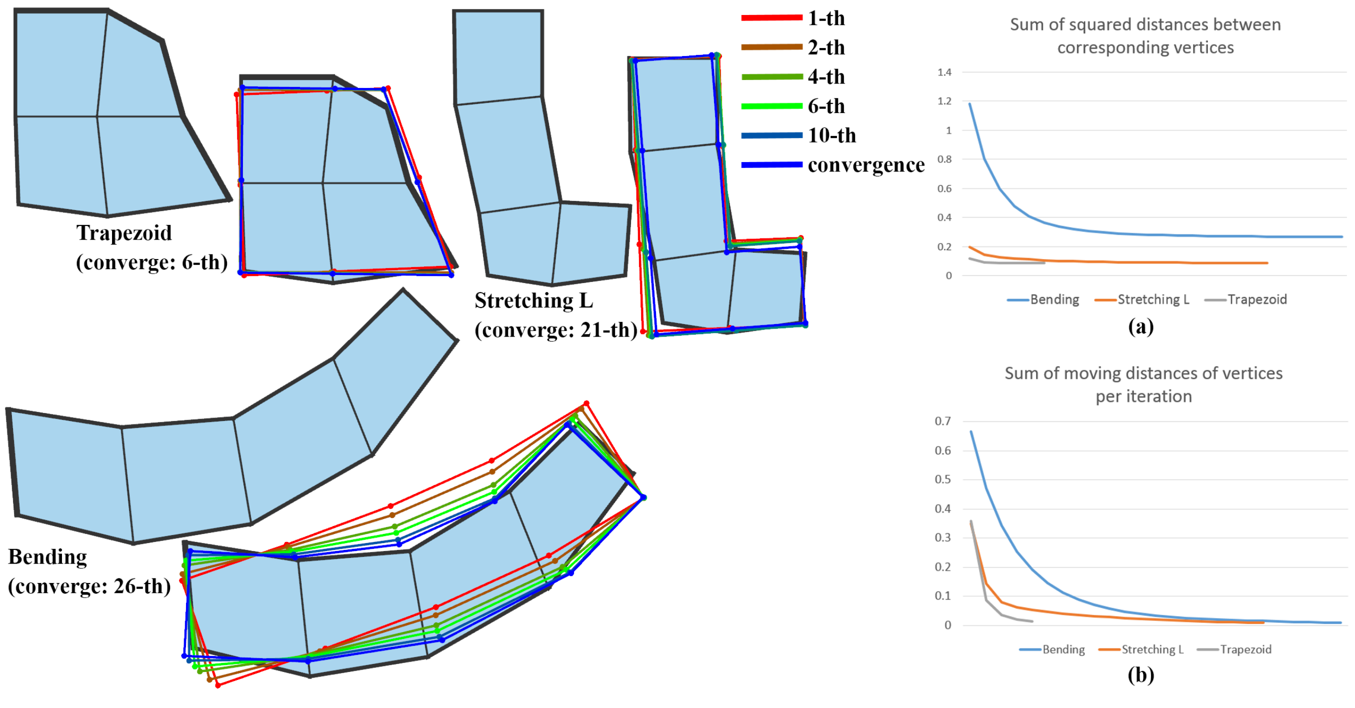

Figure 4: *Convergence analysis for the shape registration procedure. Here we show three examples: a 2x2 tile that may transform in a half-trapezoid way, an L-shaped tile where each end may stretch individually, and a 4x1 tile that may bend counter-clockwise. The error, measured as the sum of the squared distances between corresponding vertices, is monotonically decreasing, as shown in (a). Here, we take a convergence threshold of one thousandth of the average edge length for the sum of moving distances of all vertices per iteration. As shown in (b), they are also monotonically decreasing, indicating guaranteed convergences.*

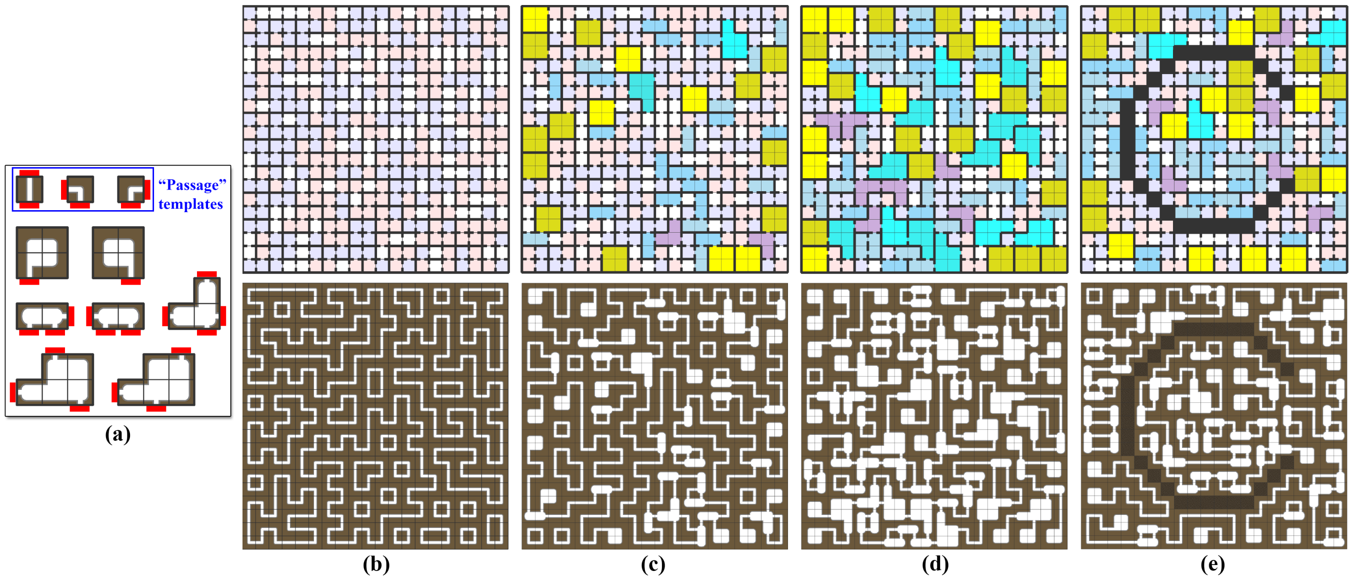

Figure 5: *Pattern design using a tile set with edge color constraints to enforce adjacency relationships between tiles. (a) There are two sets of templates: a set of templates presenting caves of predefined sizes and locations of entrances, and a set of single-quad templates to build up the passages, which are allowed to turn arbitrarily, but cannot branch out. Interesting patterns made of caves connected by passages are generated: (b) A design made up of just passages. (c) A design with caves and passages. (d) A design with higher priorities for caves. (e) A design with additional blocking constraints. Note that all the connected components can be easily connected by a post-process, e.g., iteratively digging out an additional passage between two connecting components until all components are singly connected.*

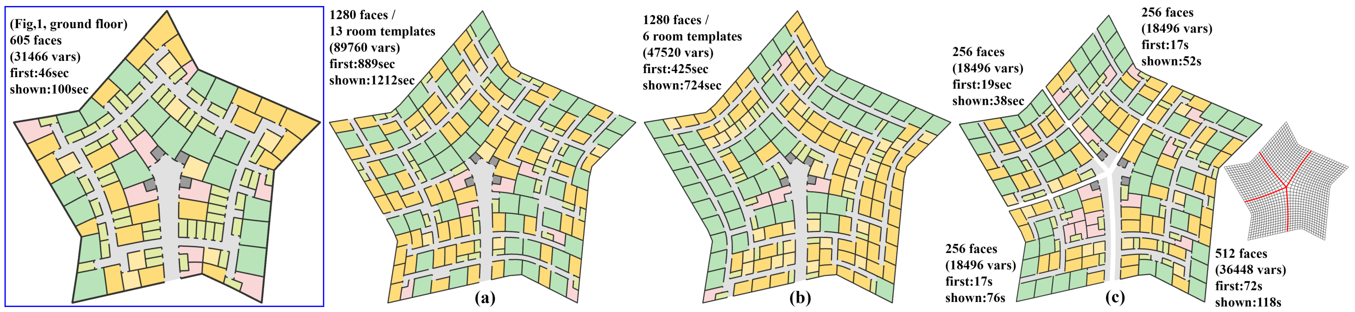

Figure 6: *Addressing the scalability issue of our approach. Left: the floorplan example of the ground floor in Figure 1. (a) The execution time grows rapidly, by 19.33 times, on a denser version of the same mesh with about twice the number of faces. To make the problem easier to solve, we can: 1) reduce the number of templates. For example, in (b), we reduce the number of room templates from 13 to 6 by discarding the room templates with alternative door locations. On the down side, the tiling looks less varied. 2) Alternatively, we can partition the mesh into sub-meshes and solve them separately. An example is shown in (c). For each result, we show the the number of faces, the number of Boolean variables, the time to get the first feasible solution, and the time to get the shown solution.*

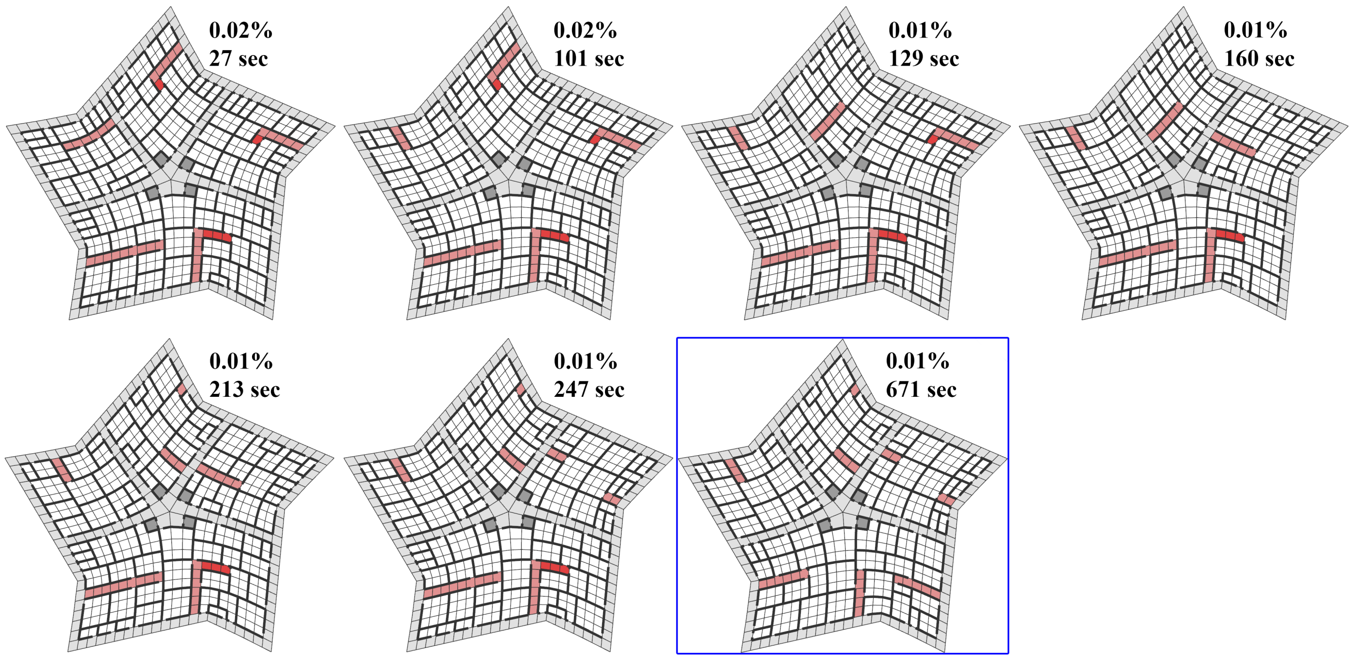

Figure 7: *Intermediate solutions found by the Gurobi solver for the floorplan of the third floor in Figure 1. Here we use a longer time limit (1000 seconds) to capture longer sequences. Each solution is noted with its closeness to the optimal solution, i.e., the gap between two objective bounds, and time. There is no newer solutions found between 671 and 1000 seconds. For difficult problems, the Gurobi solver typically finds a nearly-optimal solution, with acceptable closeness, in a relatively short time. Afterwards, it takes much longer time to further improve the solutions.*