A Graph Reduction Method for 2D Snake Problems

Jianhua Yan[1], Keqi Zhang[2], Chengcui Zhang[3], Shu-Ching Chen[1], Giri Narasimhan[1]

[1]School of Computing and Information Sciences, Florida International University, {jyan001, chens, giri}@cs.fiu.edu
[2]Department of Environmental Studies & International Hurricane Research Center, Florida International University, zhangk@fiu.edu
[3]Department of Computer and Information Sciences, University of Alabama at Birmingham, zhang@cis.uab.edu

Abstract

Energy-minimizing active contour models (snakes) have been proposed for solving many computer vision problems such as object segmentation, surface reconstruction, and object tracking. Dynamic programming which allows natural enforcement of constraints is an effective method for computing the global minima of energy functions. However, this method is only limited to snake problems with one dimensional (1D) topology (i.e., a contour) and cannot handle problems with two-dimensional (2D) topology. In this paper, we have extended the dynamic programming method to address the snake problems with 2D topology using a novel graph reduction algorithm. Given a 2D snake with first order energy terms, a set of reduction operations are defined and used to simplify the graph of the 2D snake into one single vertex while retaining the minimal energy of the snake. The proposed algorithm has a polynomial-time complexity bound and the optimality of the solution for a reducible 2D snake is guaranteed. However, not all types of 2D snakes can be reduced into one single vertex using the proposed algorithm. The reduction of general planar snakes is an NP-Complete problem. The proposed method has been applied to optimize 2D building topology extracted from airborne LIDAR data to examine the effectiveness of the algorithm. The results demonstrate that the proposed approach successfully found the global optima for over 98% of building topology in a polynomial time.

1. Introduction

Significant research effort has been placed on developing deformable models to determine a surface or a contour having optimal properties in the past decade [1]. Applications of deformable models include medical image analysis, geometric modeling, and tracking of non-rigid objects. The deformable models which are also called "snakes" or active contours were first introduced by Kass et al. in 1988 [2]. Snakes usually start with an initial one dimensional (1D) contour, two dimensional (2D) surface, or even three dimensional (3D) volume [3-5] close to a target model, and then gradually deform contour/surface while minimizing energy functions so that the resulting contour/surface best matches the target boundary/topology of the object [4, 6]. The solution to the snake problem often involves the derivation of an energy function and minimization of the energy function.

Dynamic programming is an optimization approach which finds global minima by analyzing a collection of admissible solutions. In dynamic programming, constraints are often placed on the set of allowable solutions, thus reducing the computational complexity. For example, in the case of 1D active contours, the set of admissible solutions are in fact the set of all allowed curves that connect the start point with the end point. Unlike the variational method, dynamic programming can be directly applied to the discrete grid without approximations. Amini et al. [7] devised a time-delayed discrete dynamic programming algorithm to minimize the energy for 1D active contours. The discretization of the contour energy $E(C)$ is represented by $E(C)=E_1(v_1, v_2)+ E_2(v_2, v_3)+\ldots+ E_{n-1}(v_{n-1}, v_n)$, where $C=\{v_{1,} {}_{\ldots,} v_n\}$. Each contour point v_i is allowed to only take on m possible values. Instead of using exhaustive enumeration to find the minimum of $E(C)$, a discrete dynamic programming method computes the global minimum in an efficient way. However, the dynamic programming method is inherently restricted to problems with 1D topology such as a contour [1, 7, 9]. The dynamic programming method outlined for 1D topology cannot be directly extended to the bipartite case [8]. The 2D snake problems, such as the reconstruction of surfaces cannot be solved efficiently by existing dynamic programming methods.

This paper presents an algorithm to minimize the energy function associated with 2D snakes which represent 2D surfaces with connected deformable graphs controlled by vertices and edges. The first objective of the paper is to develop the algorithm including a set of graph operations which is capable of reducing certain types of planar snakes to single vertices. The second objective is to apply the proposed method to refining building topology extracted from airborne light detection and ranging (LIDAR) measurements to examine the effectiveness of the algorithm.

The paper is organized as follows: Section 2 describes the proposed approach by first giving a formal definition of the 2D snake problem and analyzing the discretized form of the energy function, then describing the details of the proposed graph reduction algorithm, and finally proving the polynomial time complexity of the algorithm. Section 3 presents the experimental results of applying the proposed algorithm to refining building topology from LIDAR measurements, and Section 4 concludes the paper.

1-4244-1180-7/07/$25.00 ©2007 IEEE

2. The Proposed Graph Reduction Approach for 2D Snakes

The snake-based method to detect a complex graph structure (topology) involves minimizing the energy of a deformable topology. In this section, a domain-specific energy function whose minimum represents the target topology is constructed first. Then, the energy function is gradually minimized starting with an approximate topology which is usually obtained by low level image processing and pattern recognition operations. Finally, the topology corresponding to the minimum of the energy function is derived to represent the target topology. The core of this procedure is to develop an efficient algorithm for graph energy minimization. We present a graph reduction algorithm in the following subsections which is able to find global minima for certain 2D snakes.

2.1. The Search Constraint and the Cost Function

We represent the deformable topology (2D snake) with a weighted graph $G=(V, E)$. Each vertex v is associated with an uncertainty list $UL_v = \{s_v^m = (x_m, y_m) \mid m=1, 2\ldots |UL_v|\}$, which is a list of points whose distance to this vertex is less than a pre-specified distance d. The number of points in the uncertainty list represents the number of possible states of the vertex v. For each state s_v of v ($s_v \in UL_v$), there is a corresponding energy value, denoted by $EV(v, s_v)$. Correspondingly, for an edge $e = (v, w)$ connecting two vertices v and w, there are $|UL_v| \times |UL_w|$ allowable states and each state is associated with a energy value, $EE(e=(v, w), s_v, s_w)$, $s_v \in UL_v$ and $s_w \in UL_w$. For example, if v and w each has 3 possible states, the total number of states of the edge connecting these two vertices is 9. Note that the energy of an edge only depends upon the two vertices to which the edge is connected. Assuming that a list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}), s_k \in UL_k\}$ represents a state of all vertices in G, we define the cost (energy) for the deformable topologies as a sum of cost for each vertex and edge in G which is given by

$$EG(G) = \sum EV(v, s_v)_{s_v \in UL_v} + \sum EE(e = (v, w), s_v, s_w)_{s_v \in UL_v, s_w \in UL_w} \tag{1}$$

The formation of the energy functions EV and EE depends upon the application need. Minimizing the total cost (energy) generates the optimal topology that best fits the given object. Without loss of generality, we assume that there is only one minimum for EG and the corresponding state of all vertices in G is unique.

2.2. A Graph Reduction Based Implementation

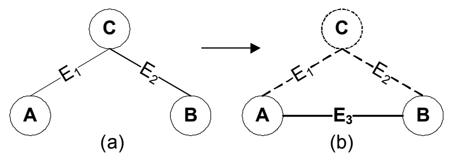
Let us consider the problem of finding the optimal topology that has the same graph structure as the given initial topology and fits best the target topology. The corresponding energy minimization is denoted by $\min\limits_{S} EG(G)$, where S is a state list for all vertices in G. A brute force implementation of finding the minimum will try all possible combinations of state lists for vertices and edges, which involves $\prod\limits_{v \in V} |UL_v|$ steps, making the time complexity exponential. It can be proven that the general 2D snake problem is NP-Complete, which means no algorithm can resolve general 2D snake problems in polynomial time. However, it is still possible to resolve a special subset of 2D problems in polynomial time. This is similar to the case of 3SAT (Boolean satisfiability problem) in which 3SAT is NP-complete, but its subset 2SAT is not NP-complete [11]. In this section, a method is proposed to derive the global minimization by progressively simplifying the graph using the following four graph reduction operations.

2.2.1 Type I operation

Given a vertex **C** which connects to two other vertices **A** and **B** via edges $\mathbf{E_1}$ and $\mathbf{E_2}$ (i.e., the degree of **C** is two) as shown in Figure 1, the vertex **C** and the two edges $\mathbf{E_1}$ and $\mathbf{E_2}$ can be reduced to a new edge $\mathbf{E_3}$ that connects **A** and **B**. The energy of the new edge $\mathbf{E_3}$ is determined by the energies of the removed vertex **C** and the edges $\mathbf{E_1}$ and $\mathbf{E_2}$ by the following equation:

$$\begin{aligned} EE(E_3 = (A, B), s_A, s_B)_{s_A \in UL_A, s_B \in UL_B} = \min_{s_C \in UL_C} \{ & EE(E_1 = (A, C), s_A, s_C) + \\ & EE(E_2 = (B, C), s_B, s_C) + EV(C, s_C)\} \end{aligned} \tag{2}$$

Equation (2) indicates that the energy of $\mathbf{E_3}$, given a pair of states s_A and s_B for the vertices **A** and **B**, is determined by the minimum sum of energies of $\mathbf{E_1}$ and $\mathbf{E_2}$ and **C**. The state s_C of **C** that minimizes Equation (2), for a given pair of s_A and s_B, is recorded in a **State Retrieval Table** as shown in Figure 1(c) after a Type I operation is performed. For example, the first row in Figure 1(c) indicates that the energy EE in Equation (2) reaches its minimum when s_C equals 3, given s_A=1 and s_B=1.



s_A	s_B	s_C
1	1	3
1	2	1
…	…	…
2	1	n
…	…	…
n	n	2

(c)

Figure 1: A Type I operation and its associated state retrieval table.

It can be proven that the reduced graph G' has the same minimal energy as that of G after a type I reduction operation is applied. The proof is omitted because of page limitation. This conclusion is also true for all reduction operations introduced in this section.

2.2.2 *Type II operation*

A Type II atomic graph reduction operation is shown in Figure 2. Given a pair of vertices **A** and **C,** if there is more than one edge connecting these two vertices in *G*, those edges can be reduced to one single edge between **A** and **C**. The energy of the new edge is the sum of the energies of all edges between **A** and **C**, as given in Equation (3). For this operation, there is no need to keep a state retrieval table because the energy of the new edge $\mathbf{E_3}$ is determined solely by the vertices **A** and **C**. No other vertex is involved in this operation.

$$EE(E_3 = (A,C), s_A, s_C)_{s_A \in UL_A,\, s_C \in UL_C} = \sum_{i=1}^{k} EE(E_i = (A,C), s_A, s_C) \tag{3}$$

Where *k* is the total number of edges that connect **A** and **C**, and *k* is equal to 2 for the example in Figure 2.

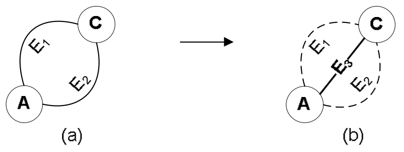


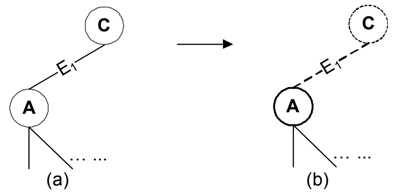
Figure 2: Type II operation.

2.2.3 *Type III operation*

A Type III atomic graph reduction operation is illustrated in Figure 3 where the vertex **C** connects to one single vertex (**A**) via one edge only (i.e., the degree of **C** is one). In this case, the vertex **C** and the edge $\mathbf{E_1}$ which connects **C** to **A** can be reduced to **A.** After reduction, the energy of **A** is updated as follows:

$$EV(A,s_A)_{s_A \in UL_A} = EV(A,s_A) + \min_{s_C \in UL_C}\{EE(E_1 = (A,C), s_A, s_C) + EV(C,s_C)\} \tag{4}$$

By Equation (4), the minimum sum of energies of **C** and $\mathbf{E_1}$, for each state of **A** (s_A), will be added to the old *EV*(*A*, s_A). Similar to the situation in Type I operation, the s_C that minimizes the sum of the energies of **C** and $\mathbf{E_1}$, will be recorded in the **State Retrieval Table** in Figure 3(c). As the states of **C** and $\mathbf{E_1}$ in the global minimization are only dependent on the state of **A**, if s_A belongs to the list $S = \{(s_1, s_2, \ldots, s_k, \ldots, s_{|V|}),\ s_k \in UL_k\}$ that minimizes the total energy of *G*, s_C must also in *S*. Therefore, the minimization of the total energy *EG*(*G*) can be reduced to minimizing the total energy of the reduced graph *G'*(*V'*, *E'*), where *V'*=*V*-C and *E'*=*E*-E$_1$.



s_A	s_C
1	3
2	1
…	…
n	2

(c)

Figure 3: A Type III operation and its associated retrieval table.

By applying the above three atomic graph reduction operations recursively, graphs with simple structures can be reduced to one single vertex $\mathbf{F}_G$, and the minimum energy of the original graph min{*EG*(*G*)} becomes $\min_{s_{F_G} \in UL_{F_G}} \{EV(F, s_{F_G})\}$. In the traditional 1D snake problem, the original graph is an opened polygon as shown in Figure 4. We can recursively apply the Type III operation to remove the leftmost vertex. Denote $n = |V|$ and $m = |UL|$. After *n*-1 operations, the original graph can be reduced to a single vertex. For each removed vertex, the state retrieval table has *m* entries and the complexity is (*n*-1)×*m*. In addition, to determine an entry in the state retrieval table, *m* calculations are needed. Thus, the total time complexity is $(n\text{-}1)\times m^2$, which is the same as reported in [7].

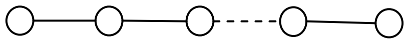
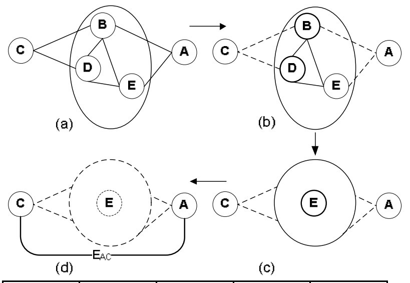


Figure 4: The topological graph *G* for a classical active contour problem.

However, in many cases, the given graph *G* cannot be reduced to one single vertex by just applying the above three atomic operations. For those cases, we proposed type IV operation which is described below.

2.2.4 *Type IV operation*

For a given pair of vertices **A** and **C**, as shown in Figure 5(a), we first find a connected subgraph $G_{AC} = (V_{G_{AC}}, E_{G_{AC}})$ in *G*, which only connects to the vertices **A** and **C**, but not to any other vertex in *G*. For example, the subgraph in an oval in Figure 5(a) is such a connected subgraph between **A** and **C**. Then the three atomic reduction operations (Types I, II, and III) are applied to subgraph G_{AC}. If G_{AC} can be reduced to a single vertex (e.g., the vertex **E** in Figure 5(c)), a Type IV operation will be applied to replace G_{AC} with a new edge connecting **A** and **C** directly (e.g., the edge E$_{AC}$ in Figure 5(d)).



s_A	s_C	s_B	s_D	s_E
1	1	3	2	1
1	2	1	3	4
…	…	…	…	…
2	1	2	3	1
…	…	…	…	…
n	n	1	2	4

(e)

Figure 5: A Type IV operation and its associated retrieval table.

The minimum energy of G_{AC}, as a portion of the minimum energy of the whole graph *G*, is not only determined by the vertices and edges of G_{AC}, but also by the pair of s_A and s_C. In a Type IV operation, the energy of each edge that connects **A** or **C** to the subgraph G_{AC} is added to that of the connecting vertex in G_{AC}. In the example shown in Figure 5(b), E_{BC}, E_{DC}, E_{EA}, and E_{BA} are such edges connecting the subgraph with **A** and **C**. Given a pair of states (s_A, s_C) for **A** and **C**, for each vertex *v* which belongs to the subgraph and connects to **A** and/or **C**, we change its energy into

$$EV(v,s_v)_{v\in V_{G_{AC}},s_v\in UL_v} = EV(v,s_v) + \sum_{(w,s_w)\in\{(A,s_A),(C,s_C)\}} EE(e=(v,w),s_v,s_w) \tag{5}$$

Let $\mathbf{F}_{G_{AC}}$ be the single vertex reduced from G_{AC} by Types I to III atomic operations for a given pair of states (s_A, s_C) for **A** and **C**. The minimum energy of $\mathbf{F}_{G_{AC}}$ is the same as that of G_{AC}, which is transferred to the energy of the new edge E_{AC} as follows:

$$EE(E_{AC}=(A,C),s_A,s_C)_{s_A\in UL_A,s_C\in UL_C} = \min_{s_{F_{G_{AC}}}\in UL_{F_{G_{AC}}}} EV(F_{G_{AC}},s_{F_{G_{AC}}}) \tag{6}$$

The state list of vertices in G_{AC} that minimizes the energy of G_{AC}, given (s_A, s_C) for **A** and **C,** is denoted by $S_{G_{AC}}$ and shown in the table of Figure 5(e). After applying the type IV operation to *G*, the problem of minimizing *EG*(*G*) can be reduced to the minimization of *EG*(*G'*) where *G'* is the reduced graph.

2.3. Finding A Reducible Connected Subgraph

Before a Type IV operation can be performed, a reducible connected subgraph G_{AC} must be found. We have developed the following algorithm which can find such a subgaph for a Type IV operation.

```
1) Set scope to 1. Initialize a set $S_{ij}$ as empty for each pair
   of vertices $i, j \in V$ (V is the set of vertices of G.)
2) For each pair of vertices i and j
   • If(scope = = 1)
      o Add the set of vertices directly connected to
        vertices i and j to $S_{ij}$.
     Else
      o Find the vertices directly connected to vertices in
        $S_{ij}$ and add them into set $S_{ij}$ except for i and j.
   • Partition $S_{ij}$ into subsets which form connected
     subgraph and examine whether any subgraph
     connects only to vertices i and j. If such a connected
     subgraph is found and can be reduced to a single
     vertex by the three atomic operations, the subgraph
     is returned and the program terminates. Otherwise,
     continue checking the next pair of vertices.
3) If all $S_{ij}$==V-{i,j}, then the program terminates.
   Otherwise, Increase scope by 1 and go to 2).
```

This algorithm gradually expands the search scope, starting with the vertices directly connected to *i* and *j*. If a reducible connected subgraph cannot be found, it adds to S_{ij} the vertices with indirect connections to *i* and *j*, and stops whenever a reducible subgraph is found. The advantage of this progressive algorithm is its efficiency – we start with the smallest search scope and expand it only when it is needed. In many cases, we can find the connected subgraph by searching only the direct neighbors of *i* and *j*. However, for *i*=1 and *j*=2 in Figure 6, we need to search those vertices indirectly connected to *i* and *j* in order to find G_{12}. In this example, S_{12} = {3, 4, 7, 8, 9, 10, 13, 14} after Step 1) and will be partitioned into S_1 = {3, 4}, S_2 = {7, 8}, S_3 = {9, 10}, and S_4 = {13, 14} at Step 2). Since S_1, S_2, S_3, S_4 connect to vertices other than *i*=1 and *j*=2, the algorithm will continue searching other pairs of vertices. Because no connected graphs for any pair of vertices can be found when *scope* ==1, the program increase *scope* by 1 and go back to step 2), where S_{12} is expanded to the set {3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}. Then S_{12} is partitioned into S_1 = {3, 4, 5, 6, 7, 8} and S_2={9, 10, 11, 12, 13, 14}. Either S_1 or S_2 can be returned as a reducible connected subgraph for *i*=1 and *j*=2 because they connect to 1 and 2 only and can be reduced to a single vertex by Type I-III operations.

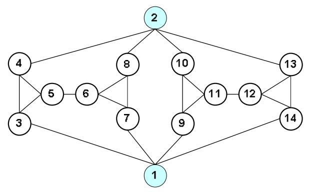


Figure 6: Another example of graph reduction.

2.4. The Algorithm of Graph Reduction

With Type I-IV operations and the algorithm for finding connected subgraphs, we have developed the following algorithm to reduce a graph *G* and derive the state list *S* = {(s_1, s_2, …, s_k, …, $s_{|V|}$), $s_k \in UL_k$} that minimizes the total energy of *G*.

Step 1: *Reduce the graph by Types I to III operations*

Without loss of generality, we scan the vertices in their numerical order, starting from vertex 1. First, each vertex in the current graph *G* is checked to see whether any of the three atomic reduction operations can be applied to that vertex. Then proper atomic reduction operations are applied to qualified vertices to reduce the current graph *G*. These operations are recorded in an array **AppliedOperations.** If the current graph *G* can be reduced to a single vertex $\mathbf{F}_G$, the procedure goes to Step 3 and constructs the state retrieval table. Otherwise, it goes to Step 2 to further reduce the graph by Type IV operations.

Step 2: *Reduce the graph by Type IV operations*

At this step, attempts are made to find one reducible connected subgraph G_{ij} by using the algorithm introduced in Section 2.3. If one is found, a Type IV operation will be performed to reduce the graph. The information related to this operation will be recorded and added to **AppliedOperations**. Then the procedure goes back to

Step 1 to reduce the current G by the three atomic operations again. If no reducible subgraphs can be found, the program will exit and report a message "This graph is irreducible."

Step 3: *Construct the state retrieval table*

Scan the operations recorded in array **AppliedOperations** from its start and apply each reduction operation to G as follows.

For each Type I/II/III atomic operation, the energy of edges and/or vertices involved will be updated according to Equations (2)-(4). The state retrieval table for each operation is constructed as well.

For each Type IV operation, we store the atomic operations used to reduce the subgraph G_{ij} into a temporary array. Given each pair of s_i and s_j, the energy of internal vertices in G_{ij} will be first updated according to Equation (5). Then we scan the atomic operations in the temporary array from the start forward, and update the energy of edges and/or vertices involved in each operation and construct the state retrieval table if applicable. After all the operations in the temporary array are performed, go to Step 4 to calculate the minimum energy of the subgraph G_{ij} and retrieve the state list minimizing the energy of G_{ij}. This state list, together with s_i and s_j, are stored as one row in the state retrieval table for that Type IV operation.

After all the operations in the array **AppliedOperations** are performed, go to Step 4 to calculate the minimum energy of G and retrieve the state list that minimizes the energy of G.

Step 4: *Determine the minimum energy and the state list minimizing the energy of graph G or a subgraph G_{ij}*

For a subgraph G_{ij}, we first determine the minimum energy of the vertex $F_{G_{ij}}$ reduced from G_{ij} and the corresponding state that minimizes the energy of $F_{G_{ij}}$. Then, the corresponding states of other vertices in G_{ij} are retrieved by scanning the temporary array backward from the end which contains all the reduction operations associated with G_{ij}. For each atomic operation, its state retrieval table is looked up and the states of the participating vertices can be determined in a retrospective way. The state list that minimizes the energy of G_{ij} is returned and the procedure goes back to Step 3.

For a graph G, we first determine the minimum energy of the single vertex $\mathbf{F}_G$ reduced from G and the corresponding state s_{F_G} that minimizes the energy of $\mathbf{F}_G$. Then the corresponding states of all the other vertices are retrieved by scanning the array **AppliedOperations** from its end. For each reduction operation, its state retrieval table is looked up and the states of the participating vertices can be determined in a retrospective way. Finally, the minimum energy value together with the state list and the **AppliedOperations** are returned as the result and the program stops.

2.5. Complexity Analysis

Let $n = |V|$ and $m' = |E|$. In Step 1, each vertex is checked to see if any atomic reduction operation (Type I/II/III operation) can be applied to it. After each atomic reduction operation, the graph shrinks by at least one vertex (operations I & III) or one edge (operation I & II). In Step 2, $C(n, 2)$ pairs of i and j may be searched to find subgraph for type IV reduction operation in the worst situation. The reduced graph can also be proven to shrink by at least one vertex or one edge. Therefore, with at most $(n+m')$ times of reduction operations, the given (reducible) graph can be reduced to one single vertex. Since each operation costs polynomial time, the whole algorithm also takes polynomial time to complete.

3. Experiment

The proposed algorithm has been applied to refining building topology from airborne LIDAR measurements to examine its effectiveness. The LIDAR technology provides an effective way to derive 2D footprints and 3D shapes of buildings by measuring building elevation directly [10]. The test data site is located at the university campus, covering 6 km^2 of low relief topography. The LIDAR data for building extraction with an average point spacing about 1 m were collected in August 2003. The buildings in the test site include residential houses, commercial buildings, and institutional buildings.

A building topology is represented by a set of connected roof plane surfaces (polygons) projected onto a 2D space, which matches our proposed "2D snake" very well. Initial footprints and internal topology of buildings are extracted from LIDAR measurements through a plane-fitting technique and regional growing algorithm. The boundaries (edges) between different roof planes are noisy, and the positions of critical corner vertices could not be located correctly in the initial footprints due to the influence of irregularly spaced point LIDAR measurements. The process for refining building topology involves adjusting the initial topology by changing the admissible states of vertices through minimizing a defined energy function. The allowable states of vertices are determined by the spatial resolution and errors of LIDAR measurements. Therefore, the refinement of building topologies from LIDAR measurements provides an excellent case to test the proposed algorithm.

The energy function for building topology refinement can be defined in any format as long as it does not violate the search constraints in Section 2.1. The energy function for this test depends upon the length of an edge and the angle between the edge and the dominant building direction. The detailed discussion about the energy function is beyond the scope of this paper since the purpose of the experiment is to investigate the effectiveness of the proposed graph reduction algorithm. The focus of the experiment is to examine whether connected building topology can be reduced by the algorithm. The percentage of building topologies that are successfully reduced by the algorithm is used to measure the effectiveness of the algorithm. A successful reduction of a building topology means that the raw topology of that building can be reduced into a single vertex using the proposed algorithm.

One data set including 67 institutional buildings in the university campus and another data set including 211 residential and commercial buildings next to the university campus have been used to test the algorithm. Our test results indicates that 210 buildings from the data set next to the university campus and 66 buildings from the data set at the university campus can be successfully reduced by the proposed algorithm. The reduction rate is over 98% in both cases. The algorithm is capable of reducing the topology of a building with a complicated shape. For example, the topology of a building in the university campus with a total of 46 vertices and 67 edges (Figure 7(a)) is successfully refined (Figure 7(b)) by the graph proposed reduction algorithm. In addition, during the graph reduction, the connected subgraphs for almost all buildings (209 of 210) are identified by only searching direct neighbors of vertex pairs using the algorithm in Section 2.3. This expedites the process of graph reduction. In our experiment, each vertex is allowed to move in a 5×5 window centered at the vertex, thus each vertex has 25 possible states. It took about 2 and 3 minutes for a PC with a 2.8 GHz processor and 2 GB RAM to complete the entire reduction process for the dataset for university campus and the dataset next to university campus, respectively.

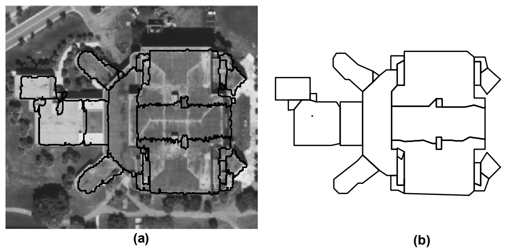


Figure 7: The raw (a) and the refined (b) topology of a complicated building at the university campus.

4. Conclusion

Discrete dynamic programming has been widely used to resolve snake problems. However, it is limited to snake problems with 1D topology and cannot handle 2D snake problems. In this research, we have demonstrated that a subset of 2D snake problems can be resolved in polynomial time. The topology of such a 2D snake problem can be reduced to a single vertex by applying the set of proposed graph reduction operations. The proposed algorithm was applied to refining 2D building topology extracted from airborne LIDAR data. Various topologies for institutional, commercial, and residential buildings have been used in the experiment. Over 98% of building topologies have been successfully reduced and their global optima have been found in polynomial time.

For those irreducible 2D snake problems, the domain–specific knowledge could be used to derive the approximate optimal solutions. For example, we can remove the least important edges in a 2D snake gradually till the remained graph is reducible. The minimum energy of the remained reducible graph plus the energies of those removed edges should be close to the global minimum energy of the original graph. This remains as our most immediate goal for future work.

References

[1] J. Montagnat, H. Delingette, and N. Ayache, “A review of deformable surfaces: topology, geometry, and deformation,” Image and Vision Computing, vol. 19, pp. 1023-1040, 2001.

[2] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: active contour models,” International Journal of Computer Vision, vol. 1, pp. 321-331, 1988.

[3] G. Subsol, J. P. Thirion, and N. Ayache, “A scheme for automatically building three-dimensional morphometric anatomical atlases: application to skull atlas,” Medical Image Analysis, vol. 2, pp. 37-60, 1998.

[4] D. Terzopoulos, A. Witkin, and M. Kass, “Constraints on the deformable models: recovering 3D shape and nonrigrid motions,” Artificial Intelligence, vol. 36, pp. 91-123, 1988.

[5] J. P. Thirion, “Image matching as a diffusion process: an analogy with Maxwell’s Demons,” Medical Image Analysis, vol. 2, pp. 243-260, 1998.

[6] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert, “Minimal surfaces based object segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, pp. 394-398, 1997.

[7] A. A. Amini, T. E. Weymouth, and R. C. Jain, “Using dynamic programming for solving variational problem in vision,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, pp. 855-867, 1990.

[8] Y. Boykov, O. Weksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, pp. 1222-1239, 2001.

[9] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, “Dynamic programming for detecting, tracking, and matching deformable contours,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, pp. 294-302, 1995.

[10] J. R. Jensen, Remote Sensing of the Environment. Upper Saddle River, NJ: Prentice-Hall, 2000.

[11] D. Lichtenstein, “Planar formulae and their uses,” SIAM Journal on Computing, vol. 11, pp. 329-343, 1982.