Optimal Reconstruction of a Surface Using A Reference Library*

Ying Xu, Victor Olman, and Edward C. Uberbacher

Informatics Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

*The submitted manuscript has been authorized by a contractor of the U.S. Government under contract No. DE-AC05-96OR22464. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.


Research was supported by the Office of Health and Environmental Research, U.S. Department of Energy under contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
Optimal Reconstruction of A Surface Using A Reference Library
(Extended Abstract)

Ying Xu, Victor Olman, and Edward C. Uberbacher
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6364

Abstract

To reconstruct (approximate) an arbitrary surface using subsurfaces (patches) from a library of surfaces in an optimal way is an interesting algorithmic problem and has many applications in image processing. This paper presents an efficient algorithm for an optimal reconstruction of a query surface using patches from a reference library of surfaces, under the constraint that the smallest patch size is above some specified value. In this algorithm, a surface is given as an integer function $f(x, y)$ over a finite 2-D grid. The algorithm partitions a query surface into patches in such a way that each patch is represented by a similar patch from a library surface, and the total difference between the query surface and the representing (composite) surface is minimized, where the boundary of a patch is not predetermined but solely determined by the optimization process. By using a minimum spanning tree-based data structure, this optimization problem can be solved efficiently. An application of this technique in computational forensics is outlined.

1 Introduction

To optimally reconstruct an arbitrary surface using patches from a library of surfaces is an interesting mathematical problem, and potentially has many applications in the domain of image processing, including image compression, object registration, recognition, and reconstruction, etc. We have recently developed an effective algorithm for an optimal reconstruction of a query surface using patches from a reference library of surfaces. Each surface is given as an integer function $f(x, y)$ defined over a finite two-dimensional grid, and all surfaces are defined on the same grid $D$. For a given query surface, the algorithm partitions $D$ into (arbitrarily-shaped) connected regions $D_1, ..., D_k$ in such a way that the patch over each $D_i$ is represented by a similar patch of a library surface, and the total difference of the representing (composite) surface and the query surface is minimized, under the constraint that the areas of all $D_i$'s have to be larger than some specified value (a parameter).

To measure the similarity between two patches, the algorithm calculates an array of (real-valued) attributes at each point $(x, y) \in D$ for each surface, which capture the surface's geometric features around $(x, y)$, e.g., the derivatives, the value of $f(x, y)$, etc. The difference (on oppositely,
the similarity) in geometry of two points in a surface is measured by the Euclidean distance between the attribute arrays of the two points. The algorithm measures the similarity between each point \((x, y)\) of the query surface and the “corresponding” point of each library surface, and then partitions \(\mathcal{D}\) into (connected) regions so that each region is “consistently” (possibly with a few exceptions) more similar to one library surface than the others. As a result, the query surface is partitioned into patches with each one having one most similar patch in the reference library. These library subsurfaces form the representing (composite) surface of the query. Figure 1 illustrates the basic idea of the algorithm.

To implement this algorithm efficiently, we have formulated the two-dimensional grid partitioning problem, as outlined above, as a tree partitioning problem through using a minimum spanning tree-based data structure. The basic idea of this implementation can be described as follows: (1) We first construct a weighted graph with each node representing a point \((x, y) \in \mathcal{D}\) and each edge representing the adjacency relationship between two points of \(\mathcal{D}\). The array of similarities of each point (its geometric attributes) of the query surface to its “corresponding” points in the library surfaces determines the characteristic of the point. The weight of an edge connecting two adjacent points...
points is determined proportionally to the difference of their characteristics. (2) Then we build a minimum spanning tree of the graph. Intuitively, a minimum spanning tree is obtained from the graph by removing edges with large weights (edges connecting points with significantly different characteristics). Based on this observation, the goal of partitioning the 2-D grid into regions with points having similar characteristics can be essentially achieved through partitioning the minimum spanning tree. (3) A tree can be partitioned into subtrees with nodes having similar characteristics using a fast dynamic programming algorithm.

We have applied this algorithm to a face reconstruction problem from Human skull data (3-D MRI images) as part of the computational forensic effort [1] at Oak Ridge National Laboratory. One goal of this project is to predict the facial structure of a given Human skull, based on a database of 3-D Human skull images and their corresponding face images. Preliminary application results are quite promising by using this algorithm.

2 Reference-based Surface Reconstruction

This section formally defines the reference-based surface reconstruction problem and presents a dynamic programming algorithm for solving the problem. In essence, this problem generalizes the image segmentation problem [2, 3] since in an image segmentation problem, the goal is to partition an image into connected regions of similar textures or similar colors/gray-levels, while the goal here is to partition a surface into patches, each of which consists of points that are consistently similar (in geometry) to a patch of the same library surface.

2.1 Problem formulation

Let \( f_0() \) be an integer function over \( D \), representing the query surface; and \( \{f_1(),...,f_n()\} \) be a set of integer functions also over \( D \), representing the reference library of surfaces, where \( D = I_N \times I_N \) with \( I_N \) representing the first \( N \) natural numbers. Let \( A_i(u) = \{A_1^i(u),...,A_p^i(u)\} \) represent an array of \( p \) attributes\(^1\) of \( f_i() \) at point \( u = (x,y) \), for \( i \in [0,n] \).

To directly solve the problem of partitioning the 2-D grid \( D \) into regions consisting of connected points having similar attributes is computationally intractable since the partitioning a 2-D object to optimize any non-trivial function is intrinsically difficult. To overcome this difficulty, we have formulated the reference-based surface reconstruction problem in terms of a tree partitioning problem, which we give now.

We first define a weighted graph \( G = (V,E) \), where the node set \( V = \{ \text{all points of } D \} \) and the edge set \( E = \{(u,v) | u,v \in V \text{ and } distance(u,v) \leq 2 \} \), where \( distance() \) is the Euclidean distance between \( u \) and \( v \). We define the (dis)similarity of \( f_0() \) and \( f_i() \) at a point \( u = (x,y) \) as

\[
C_i(u) = \rho(A_0(u), A_i(u)),
\]

\(^1\)These attributes are intended to capture the geometric shape of \( f_i() \) around point \( (x,y) \), and it may include the value of \( f_i(x,y) \), the first-order derivatives along the \( x \)- and \( y \)-axis, etc.
where \( \rho(X,Y) \) is any function of “similarity” between vectors \( X \) and \( Y \). The function we have used in our application is \( \rho(A_0(u), A_i(u)) = \sqrt{\sum_{j=1}^{p}(A_0^j(u) - A_i^j(u))^2} \). We call \( C(u) = (C_1(u), ..., C_n(u)) \) the characteristic of \( f_0() \) at point \( u \), which gives the spectrum of similarities of \( f_0() \) to \( \{f_1(), ..., f_n()\} \) at point \( u \in \mathcal{D} \).

Using the characteristic of each point, we can define the weight of each edge of \( G \), connecting two adjacent nodes \( u \) and \( v \) as follows:

\[
w(u, v) = \sigma(C(u), C(v)), \tag{2}\]

where \( \sigma(X,Y) \) can be any function that increases as the “difference” between \( X \) and \( Y \) increases. In our implementation, we have defined \( \sigma() \) as follows. Let \( s(u,i) \) be the order position of \( C_i(u) \) in the sorted list (in the non-decreasing order) of \( (C_1(u), ..., C_n(u)) \), and \( t(u) \) be the index \( i \) such that \( C_i(u) \) is the first in this sorted list. We define

\[
\sigma(u, v) = s(u, t(v)) \times C_{s(u)}(u) + s(v, t(u)) \times C_{s(v)}(v). \tag{3}\]

Note that a large weight of an edge indicates that the two connected points have significantly different attribute values, i.e., they should probably belong to different partitioned regions. This suggests that if we remove some of the edges with large weights, and then partition the subgraph (with edges of large weights being removed) into connected subgraphs with similar attributes, the partitioning result should be quite similar to the partition results on graph \( G \), which corresponds to the original 2-D grid partitioning problem. We have used the minimum spanning tree of graph \( G \) to implement this idea. A tree structure is simple enough to facilitate a fast partitioning algorithm, and also general enough to capture the necessary information for the purpose of region partitioning. A similar idea has been used in our previous work on gray-level image segmentation [4].

A minimum spanning tree of a weighted graph can be found using greedy methods [5], as illustrated by the following strategy: the initial solution is a singleton set containing an edge with the smallest weight, and then the current partial solution is repeatedly expanded by adding the next smallest weighted edge (from the unconsidered edges) under the constraint that no cycles are formed until no more edges can be added. For the above defined planar graph \( G \), a minimum spanning tree can be constructed in \( O(|V| \log(|V|)) \) time and in \( O(|V|) \) space.

We now define the reference-based surface reconstruction problem. Given is a tree \( T \), representing the minimum spanning tree of graph \( G \), as defined above. Each node \( u \) of the tree represents a point in \( \mathcal{D} \), and has a list of \( p \) attributes \( C(u) \), as defined above. We want to find a partition \( \{T_1, ..., T_k\} \) of \( T \) and a mapping \( M \) from \( \{T_1, ..., T_k\} \) to \( \{1, ..., n\} \) (representing the \( n \) functions \( f_i() \)) such that each \( T_i \) is a subtree of \( T \) and the following function is minimized:

\[^2\text{Another possible way to define the weight of an edge (u,v) is to use the cosine value of the angle between the two vectors (1/C_1(u), ..., 1/C_n(u)) and (1/C_1(v), ..., 1/C_n(v)). The general idea is that in a smooth surface, adjacent points belonging to the same subsurface should have very similar characteristics, while points belonging to different subsurfaces (boundary points) may not. A “good” } \sigma() \text{ function should capture this intuition.}\]
\[
\text{minimize } \sum_{i=1}^{k} \sum_{u \in T_i} \rho(A_0(u), A_i(u))
\]

subject to: \[ \|T_i\| \geq S, i \in [1, k]. \]

where \( k \) is not pre-determined, \( \| \cdot \| \) represents the cardinality of a set, and \( S \) is a parameter.

In our application to the face reconstruction problem, we have solved a variation of this problem. There we want to minimize a slightly different objective function, given as follows.

\[
\text{minimize } \sum_{i=1}^{k} \sum_{u \in T_i} \min_{u' \in \mathcal{N}(u)} \{ \rho(A_0(u'), A_i(u')) \}
\]

subject to: \[ \|T_i\| \geq S, i \in [1, k], \]

where \( \mathcal{N}(u) \) represents a specified neighborhood of a point \( u \).

### 2.2 A dynamic programming algorithm

This subsection presents a dynamic programming algorithm for the minimization problem (4) defined in Section 2.1. The algorithm is general enough for any (dis)similarity function \( \rho() \), which is computable. The algorithm can also be easily extended to solve the more general minimization problem (5) of Section 2.1.

The algorithm first converts the tree (minimum spanning tree) into a rooted tree by selecting an arbitrary node as the root. Hence the parent-child relation is defined. For each tree node, the algorithm constructs a minimum solution to (4) on the subtree rooted at the node, based on the minimum solutions on the subtrees rooted at its children. The algorithm repeatedly extends a partial solution in such a bottom-up fashion and stops when it reaches the root.

We assume that the nodes of \( T \) are labeled consecutively from 1 to \( \|T\| \) with the tree root labeled as 1. We use \( T^i \) to denote the subtree rooted at node \( i \). Note the difference between \( T^i \) and \( T_i \), the latter denoting one of the partitioned subtrees. Let \( \text{score}(i, k, p) \) denote the minimum value of the objective function of (4) among all possible partitions and all possible mappings on subtree \( T^i \), which satisfy the required constraint except that the partitioned subtree containing \( i \) has at least \( k \) nodes and is mapped to a fixed \( f_p \), where \( k \in [0, S] \) and \( p \in \{1, ..., n\} \). If we define \( \text{scores}() \) to be \( +\infty \) for all undefined triples \((i, k, p)\) (note \( \text{scores}() \) is not defined everywhere by the above definition) we have the following equations, which essentially describes how an optimum partition on a subtree rooted at a node is related to the optimum partitions on subtrees rooted at this node’s children. If \( i_1, i_2, ..., i_n \) are the children of node \( i \), and \( 1 \leq k \leq S \), we have

\[
\text{score}(i, k, p) = \min_{k = \sum_{j=1}^{n} k_j, \sum_{j=1}^{n} \text{score}(i_j, k_j, p) + \rho(A_0(i), A_p(i))} \quad \text{when } \|T^i\| \geq S,
\]

\[
\text{scores}(i, k, p) = \begin{cases} \sum_{j \in \mathcal{A}(i)} \rho(A_0(j), A_p(j)), & \|T^i\| = k \\ +\infty, & \|T^i\| \neq k \end{cases} \quad \text{when } \|T^i\| < S
\]

\[ (6) \]
where \(d(i)\) is the set of all \(i\)'s descendants, and \(i\) is defined to be \(\in d(i)\). Note that \(\min_p \text{score}(1, S, p)\) is a minimum solution of (4), where 1 represents the tree root.

Based on the recurrences of (6), we can solve the optimization problem (4) by calculating \(\text{score}()\) for each tree node in a bottom-up fashion using the recurrence, and stopping at the tree root. We omit further details. This algorithm can be implemented in \(O(||T|| - S).S^2n)\) time and \(O(||T||.S.n)\) space if \(\rho()\) can be computed in \(O(1)\) time, where \(S\) is smallest allowed subtree size.

3 Applications and Discussions

We have applied the reference-based surface reconstruction algorithm in the face reconstruction project at Oak Ridge National Laboratory. The goal is to reconstruct faces from Human skull data, based on a database of 3-D skull and face images. As a preliminary test, we have applied the algorithm on both 2-D images and 3-D images to examine how well a query image is represented by a database of images. We now show two test examples.

In the first test, we have used a set of eight Human face images from the US Army FERET Database [6], after some pre-processing including some approximate alignment and normalization for lightings. In this test, we have used each of the eight images as a query image and the rest seven as the reference images to reconstruct the query image. With seven images (128 x 128) in the reference library, each reconstruction takes about 1 CPU minute on a SPARC 20 workstation. Figure 2 gives the test results. Note that the boundaries of patches can be easily seen in the reconstructed composite images in Figure 2 (b).

In the second test, we have used seven 3-D face images (MRI images). We have arbitrarily picked one as the query image and rest six as the reference library. All the images are approximately aligned and scaled. Because of the generality of our algorithm, 3-D images are treated the same way as 2-D images except possibly using slightly different set of attributes. The reconstruction result is given in Figure 3.

In conclusion, we have developed an effective algorithm to reconstruct a query surface using patches from a reference library of surfaces, and have applied the algorithm in a face reconstruction project, using a database of 3-D skull and face images. The preliminary results are very promising.

Acknowledgements

This research was supported by the United States Department of Energy, under contract DE-AC05-84OR21400 with Lockheed Martin Energy Systems, Inc.

The authors want to thank Dr. Dan Tufano of Oak Ridge National Laboratory for many helpful discussions related to the work presented here. We also want to thank Mr. Manesh Shah of Oak Ridge National Laboratory for providing us the processed FERET face images. The MRI images used in Figure 3 are derived from anonymous digital data provided by the Radiology Department of the University of Tennessee, Knoxville.
References


Figure 2: Images in (a) are face images (128 x 128) from the FERET database after some approximate alignment and normalization for lightings. Each image in (b) is a reconstructed composite image using the other seven FERET images.
Figure 3: (a) is a library of six Human face images (MRI images). The image on the right in (b) is a side-view of a 3-D query image, and the image on the left is the reconstructed composite image (from the same view) using the library of six images.