Shape Reconstruction with Topological Priors

by

Ying Zheng

Department of Computer Science
Duke University

Date: __________________________
Approved:

______________________________
Herbert Edelsbrunner, Supervisor

______________________________
Carlo Tomasi

______________________________
Xiaobai Sun

______________________________
John Harer

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2012
Abstract

Shape Reconstruction with Topological Priors

by

Ying Zheng

Department of Computer Science
Duke University

Date: ______________________
Approved:

Herbert Edelsbrunner, Supervisor

Carlo Tomasi

Xiaobai Sun

John Harer

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University 2012
Abstract

We show topological priors play an important role in solving the inverse problem of shape reconstruction. We classify the applications into 1D, 2D, and 3D cases:

In 1D, we show that the persistent extrema of the curvature function of a closed curve are useful for shape simplification. In 2D, we study how to label a scene into multiple tiers to approximate the actual scene layout. We use the number of extrema as a topological prior to bound the complexity of the shape of tiers and study 2D labeling under symmetry shape priors. In 3D, we recover the detailed 3D root shape using multiple 2D images. Three novel ideas are presented. First, we propose the use of harmonic images for background subtraction. Second, we develop the regularized visual hull to preserve the details of an example image in reconstruction. Third, we enforce the topological connectedness by an efficient algorithm that is inspired by the recent development of persistent homology.

Computational efficiency is emphasized throughout the thesis. We show that 1D topological persistence can be computed in $O(n)$ time on a closed curve of $n$ nodes. For 2D tiered labeling, we give an approximation algorithm to compute it in $O(nK)$ time for $K$ tiers on an image of $n$ pixels. For 3D root reconstruction, we accelerate the computation using oct-trees and minimal spanning trees. With these ingredients, it takes only a few seconds to reconstruct a detailed root shape from 40 images of resolution $1600 \times 1200$ on a laptop.
# Contents

Abstract iv

List of Figures vii

Acknowledgements ix

1 Introduction 1

2 Topological Persistence and Shape Simplification 4
   2.1 Functions on a Closed Curve 5
   2.2 Topological Persistence 7
   2.3 The Algorithm 8
      2.3.1 Analysis 9
   2.4 Shape Simplification 10
      2.4.1 Measuring Local Convexity 10
      2.4.2 Persistence-Based Simplification 11
      2.4.3 Curve Approximation via Circular Arcs 11
      2.4.4 Results 12

3 Tiered Labeling with Topological Priors 14
   3.1 Literature Review 15
   3.2 Tiered Labeling 18
      3.2.1 1D Tiered Labeling 19
      3.2.2 2D Tiered Labeling 20
3.3 Symmetric Object Labeling ........................................... 23
3.4 Topological Smoothness ............................................... 25
3.5 Experiments and Demonstrations ................................. 27

4 Detailed 3D Root Reconstruction ................................. 34
  4.1 Literature Review .................................................. 36
  4.2 Harmonic Background Subtraction ............................ 38
  4.3 Regularized Visual Hull .......................................... 40
  4.4 Repairing Connectivity .......................................... 44
  4.5 Experiments ....................................................... 46

5 Conclusions .............................................................. 51

Bibliography ................................................................. 53

Biography ................................................................. 58
# List of Figures

2.1 Local extrema on a closed curve .............................................. 5  
2.2 Persistence of local extrema of a noisy function ........................... 6  
2.3 Left and right persistence of local extrema ................................. 8  
2.4 Illustration of Algorithm 2 .................................................... 10  
2.5 Example results of shape simplification .................................... 13  
3.1 Tiered labeling under symmetry and topological smoothness priors . 15  
3.2 1D tiered labeling ................................................................. 18  
3.3 Convert multi-tier labeling to binary labeling .............................. 20  
3.4 Unfolding object labels by vertical decomposition ...................... 22  
3.5 Folding cost matrix for symmetric labeling ............................... 24  
3.6 Cartoon city skyline ............................................................... 25  
3.7 Example results of tiered labeling .......................................... 28  
3.8 Comparison of details in tier boundaries ................................... 29  
3.9 Example multi-tiered labeling .................................................. 31  
3.10 Symmetric object labeling ...................................................... 32  
3.11 Preserving sharp transitions in tier boundaries ........................... 33  
4.1 Plant root imaging system ....................................................... 35  
4.2 Close up image of two roots in a gel container ........................... 36  
4.3 Five views of reconstructed 3D shapes of two roots ..................... 37  
4.4 Harmonic background subtraction .......................................... 40
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Comparisons of background subtraction methods</td>
<td>41</td>
</tr>
<tr>
<td>4.6</td>
<td>Problems of visual hull</td>
<td>43</td>
</tr>
<tr>
<td>4.7</td>
<td>Harmonic background</td>
<td>48</td>
</tr>
<tr>
<td>4.8</td>
<td>Regularized visual hull preserves details</td>
<td>49</td>
</tr>
<tr>
<td>4.9</td>
<td>Example reconstructed 3D plant roots</td>
<td>50</td>
</tr>
</tbody>
</table>
Acknowledgements

I would like to thank my advisor, Professor Herbert Edelsbrunner, for guiding me through the Ph.D. program. He always encourages me to pursue my own ideas, shares me with his wonderful thoughts, gives me generous support whether I decide to stay at Duke or IST Austria, and points me toward right directions. He was extremely supportive when I had to start my job at Google earlier than planned. He is a great advisor with amazing patience and tremendous support for students. I feel extremely lucky to have Herbert as my advisor.

I would like to thank Professor Carlo Tomasi for sharing with me his broad knowledge in computer vision, and for his invaluable suggestions on improving my papers for publication at top venues. I enjoyed every bit of my discussion with him, both in class and in his weekly reading group. I thank Professor Philip Benfey for his generous support and many friendly discussions. He is always willing to listen, ready to help, and offers many insightful suggestions on how to improve my work. I thank Professor Xiaobai Sun for discussing with me about my thesis and study, and for offering her advice both in class and in person. I thank Professor John Harrer for being my committee member and for his suggestions on how to improve this thesis.

I enjoyed participating in the RSA meeting, and would like to thank my group members and peers for their invaluable help and friendship. Special thanks to Anjali S. Iyer-Pascuzzi, Christopher N. Topp, Olga Symonova, Sophie Fang, and Joshua S. Weitz. I enjoyed talking to Chao Chen, Michael Kerber, Paul Bendich, and Brittany
Fasy at IST Austria. Special thanks to Chao Chen for hanging and presenting our poster at ICCV’11 in Spain, Barcelona.

I am grateful to my parents for their unconditional support, encouragement, and love. Above all, I thank my husband, Steve, for his company and love.
We study shape reconstruction under various topological priors in 1D, 2D, and 3D cases. In 1D, the input is a noisy closed curve representing the boundary of a 2-dimensional shape. The objective is to produce a clean shape boundary that is intuitively simple and preserves the gist of the shape. In 2D, the input is an image of a typical indoor or outdoor scene. The task is to segment and label the image into multiple tiers (e.g. front, middle, back) and derive the shape of each tier. In 3D, the input is a series of calibrated 2D images of plant roots growing in a gel container. The task is to recover the fine details of the 3D plant root, i.e., long, thin, and twisted branches of the root system. None of the three tasks are easy because they are inherently inverse problems that are ill-posed unless additional priors are imposed. The main questions are:

- How to quantize the noise of a shape boundary?
- How to derive the shape of each tier in an image?
- How to ensure that the recovered 3D root shape respects details and other properties?
We explore the use of topological priors in solving these problems in a practical way Zheng et al. (2011, 2012b,a). First, we study the topological persistence of a function defined on a closed curve. It is typical that a noisy function has lots of local extrema, most of which are not important to the description of the shape. We use topological persistence to measure the resilience of each local extremum to perturbations, and remove those local extrema that are not persistent enough. We then use the remaining extrema to reconstruct the shape boundary and demonstrate cleaner results over the noisy inputs.

Second, in 2D tiered labeling, we explore two topological priors: the number of local extrema of a scene boundary, and symmetry. We use the number of local extrema to quantify the scene complexity. For instance, when viewed in perspective, the floor boundary that separates the ground and the wall would first increase monotonically and then decrease monotonically, leaving at most one local maximum. Similarly, the boundary curve that separates the sky from the building would first decrease monotonically and then increase monotonically, leaving at most one local minimum. Enforcing such a shape prior would enhance the tiered labeling quality, as we demonstrate in experiments. We also explore the use of symmetry in labeling objects such as human faces, and more general symmetric patterns.

Third, in 3D plant root reconstruction, we guarantee the topological connectedness of the reconstructed shape in voxel representation. In addition, we develop the regularized visual hull to ensure that the 3D shape preserves details of an example image, and that the result is insensitive to calibration inaccuracy and local jittering. We also propose to model the gel background as a harmonic patch for high quality background subtraction, which is essential for extracting the details of plant root systems. Combined together, we demonstrate a highly effective processing pipeline for recovering the true 3D plant root shape from multiple 2D images in an unsupervised way.
Computational efficiency is emphasized throughout the thesis. For the shape simplification, we develop an $O(n)$ time algorithm to compute the topological persistence of all local extrema on a curve of $n$ nodes. For tiered labeling, we give an approximation algorithm that runs in $O(nK)$ time to find $K$ tiers on an image of $n$ pixels. For the 3D root reconstruction, we adopt the oct-tree data structure to accelerate the computation at hand, and enforce the connectedness among voxels via a simple minimal spanning tree algorithm. It takes about 3 seconds to reconstruct a detailed root shape from 40 images of resolution $1600 \times 1200$ on a quad-core laptop.
Topological persistence, originally developed by Edelsbrunner et al. (2002), measures how resilient the local extrema of a function are to perturbations of the function values. The higher the topological persistence of a critical point, the more likely the point is to survive a perturbation of the shape or function. Because of this, persistence is also intimately related to the notion of stability in control theory.

Topological persistence and its computation have been studied in a general setting by Edelsbrunner and Harer (2008); Edelsbrunner et al. (2002) and applied in biology to measure the periodicity of gene expression Dequeant et al. (2008); Cohen-Steiner et al. (2010). In this chapter, we focus on functions defined on a closed curve. This restriction has two benefits: First, it allows for a simple, self-contained definition of topological persistence. Second, it leads to a simple and efficient algorithm, whose linear-time performance is asymptotically optimal and improves – for the 1D case – upon the complexity of more general algorithms described in the literature.

We show that calculating topological persistence is useful for curve simplification, a.k.a shape abstraction by Hoppe (1996); Popovic and Hoppe (1997); Garland
2.1 Functions on a Closed Curve

A closed curve is a non-self-intersecting, continuous loop in the 2-dimensional plane. For images, we discretize a closed curve as a sequence of pixels, \( C = \langle p_0, ..., p_{n-1} \rangle \), with \( p_i \neq p_j \) for any \( 0 \leq i \neq j < n \). Any cyclic permutation of the sequence and Heckbert (1997); Agarwal et al. (2005); Hershberger and Snoeyink (1994); Douglas and Peucker (1973) in the literature of computer graphics and computer vision. Curve simplification is useful for noise removal, efficient storage, and a simplified representation of shape contours as a basis for further reasoning (e.g. DeMenthon et al. (1998); Etou et al. (2004); Mi et al. (2009)). Our algorithm yields high compression rates through very fast computation. Our experiments are mainly proof-of-concept, but hint at the usefulness of our method in several applications, including shape compression and de-noising; stylized editing of hand-drawn curves; and the simplification of object boundaries in images.
Figure 2.2: Not all the local maxima of $f$ are informative and most of them do not survive small perturbations of the function. In one dimension, topological persistence measures the stability of local extrema of $f$. The solid circles are the most persistent maxima, and the empty circles are not persistent ones.

represents the same curve. Given a function $f: C \to \mathbb{R}$ (Figure 2.1), a total ordering $\prec$ of the pixels in $C$ is:

**Definition 1** (Order $\prec$). We say $p_i \prec p_j$ if (1) $f(p_i) < f(p_j)$, or (2) $i < j$ and $f(p_i) = f(p_j)$.

Thus, index values break ties between pixel values, so that local maxima and minima of $f$ can be defined unambiguously. Since the curve $C$ is a closed loop, it is convenient to use modular indices: for any integer $0 \leq i < n$ and for any integer $x$, we define $p_{i+x} \triangleq p_{(i+x) \mod n}$ as the cyclic shift of index $i$ by $x$.

**Definition 2** (Local Extrema). We say that $p_i \in C$ is a local maximum of $f$ if $p_{i-1} \prec p_i$ and $p_{i+1} \prec p_i$; $p_i$ is a local minimum of $f$ if $p_i \prec p_{i-1}$ and $p_i \prec p_{i+1}$.

By our definition, any curve $C$ has at least one local extremum. The number of extrema is large if the function $f$ contains noisy measurements (Figure 2.2). For example, perturbing a constant function arbitrarily but within a small magnitude yields multiple local extrema, none of which are informative. This justifies ranking the stability of each extremum through the notion of topological persistence.
2.2 Topological Persistence

In the 1-dimensional case, topological persistence measures the lifetime of a local maximum or a local minimum of \( f \). Since the local maxima of \( f \) become the local minima of \( -f \), we only discuss how to define and compute the persistence of local maxima.

The notion of persistence can be understood from many perspectives and the most general version is introduced through homology theory. In this chapter, we find it simpler to start with a notion related directly to a type of stability:

**Definition 3** (\( \delta \)-Stability). The point \( p_i \in C \) is \( \delta \)-stable if there exist \(-n < l < 0 < r < n \) and \( r - l \leq n \) such that \( p_{i+x} < p_i \) for any \( x \in [l, r] \) and \( f(p_i) \geq \max\{f(p_{i+l}), f(p_{i+r})\} + \delta \).

In words, the function \( f \) at a \( \delta \)-stable maximum \( p_i \) is greater than its values in an interval around \( p_i \), and greater by a margin of at least \( \delta \) than the values at the endpoints of that interval. Clearly, all local maxima of \( f \) are at least 0-stable. We use \( \delta \)-stability to define topological persistence:

**Definition 4** (Topological Persistence). The persistence of \( p \in C \), denoted \( \varrho(p) \), is the maximal \( \delta \) under which \( p \) is \( \delta \)-stable. In other words, \( p \) is \( \varrho(p) \)-stable but is not \( (\varrho(p) + \varepsilon) \)-stable for any \( \varepsilon > 0 \).

A physical picture gives an intuitive notion of persistence: The blue dot on the left in Figure 2.3 is the lowest point that a ball that rolls under gravity and without friction reaches when released from just to the left of the red dot. The blue dot on the right is the lowest point if the ball is released just to the right of the red dot. Persistence is the smaller of the two changes in altitude, each measured as positive.

From a signal processing point of view, the higher the persistence of a local maximum \( p \), the more stable \( p \). Persistence is a global quantity for some local
2.3 The Algorithm

We present a linear-time algorithm to compute the persistence for all the local maxima of a function $f$ on a closed curve. First, note that only maxima and minima of $f$ are needed, since the values of the remaining pixels do not affect the computation. Second, according to the physical picture of persistence, we can compute the left and right persistence of each maximum separately, and then take the minimum of the two. Therefore we only focus on how to compute left persistence in this section. The computation of right persistence is analogous.

In order to compute, say, left persistence, we traverse the curve $C$ clockwise starting from the global maximum of $f$. Each time, we check whether a pixel is a maxim, since its computation can possibly involve the entire domain of the curve. Local extrema can be ranked by persistence, and low-persistence extrema can be removed, resulting in a simplification of the curve. Section 2.4 shows one way to “remove” low-persistence extrema. We turn to the computation of persistence first.
local maximum or minimum and update the left persistence of any maximum as it is visited. We use a stack $S$ to store local maxima by decreasing function values (when read bottom to top), and bookmark the minimal values between consecutive local maxima in another stack $V$. See Algorithm 1 for details.

**Algorithm 1** Compute the persistence of local maxima of $f$.

Set persistence $\varrho(p) \leftarrow +\infty$ for each local maximum of $p$;
Set the persistence of the global maximum $p_i$: $\varrho(p_i) \leftarrow \max_{q \in C} f(q) - \min_{q \in C} f(q)$;
for $\Delta \in \{+1, -1\}$ do
    Push the global maximum $p_i$ of $f$ to an empty stack $S$;
    Initialize an empty stack $V$ to hold local minima of $f$;
    for $j = 1$ to $n - 1$ do
        Move to the next point: $v \leftarrow p_{s(i,j\Delta)}$;
        if $v$ is a local minimum of $f$ then
            $V.push(v)$;
        else if $v$ is a local maximum of $f$ then
            $u \leftarrow V.top()$;
            while $S.top() \prec v$ do
                $S.pop(), V.pop()$;
                if $V.top() \prec u$ then
                    $u \leftarrow V.top()$;
            end if
        end if
    end for
    Update: $\varrho(v) = \min\{\varrho(v), f(v) - f(u)\}$;
    $S.push(v), V.pop(), V.push(u)$;
end for

The step of the process for updating persistence is depicted in the first two panels of Figure 2.4. Note that $push()$, $pop()$, and $top()$ are the standard stack operations.

2.3.1 Analysis

Although the algorithm has a while loop, each pixel appears in either stack at most twice. Therefore, the complexity of the algorithm is $O(n)$, where $n = |C|$ is the length of the curve. The correctness of the algorithm follows from the fact that whenever a local maximum is pushed into the stack, its left (or right) persistence is correctly computed by finding the lowest local minimum to the left (or right) side of it. This is realized by the while loop in Algorithm 1.
2.4 Shape Simplification

Topological persistence is useful for shape simplification. In particular, we use persistence to determine the stable extrema in the curvature of $C$, and we approximate the curve between consecutive extrema by circular arcs. This yields a cleaner, more compact, and typically faithful representation of the original curve, in which salient cusps are preserved. Our method requires three steps: computation of local convexity, removal of low-persistence extrema of convexity, and approximation of intervals between high-persistence extrema with circular arcs.

2.4.1 Measuring Local Convexity

Let $p, q, r$ be three consecutive points on the curve $C$, traversed clockwise. We define the local convexity $f(q)$ at $q$ as the clockwise rotation angle (in $[0, 2\pi]$) between the rays $\overrightarrow{qp}$ and $\overrightarrow{qr}$. We say that $q$ is convex if $f(q) < \pi$ and concave if $f(q) > \pi$. For greater resilience to noise, we use a method similar to the Parzen window to estimate the convexity: we first estimate the convexity at $p_i$ from the triples $(p_{i-1}, p_i, p_{i+1})$, $(p_{i-2}, p_i, p_{i+2}), \ldots, (p_{i-k}, p_i, p_{i+k})$, and then average the results with Gaussian weights.
(smaller weights for more distant points).

2.4.2 Persistence-Based Simplification

We preserve only the persistent extrema (local maxima and minima) of the local curvature function \( f : C \to [0, 2\pi] \) in order to give a compact representation of the shape. Specifically, we preserve a local maximum \( p \in C \), if and only if:

\[
\rho(p) \geq \max \left\{ \alpha \left[ \max_{q \in C} f(q) - \min_{q \in C} f(q) \right], \beta \right\}
\]

where \( \alpha, \beta \in [0, 1] \) are parameters and \( \max_{p \in C} f(p) - \min_{p \in C} f(p) \) is the persistence of the global maximum of \( f \). The first term in the braces specifies that the persistence of the local maximum \( p \) must be large enough compared to the persistence of the global maximum, and the second term, \( \beta \), handles degenerate case properly. A degenerate curve has almost equal curvature everywhere (e.g. a not so perfect circle). In our experiments, we fix \( \alpha = \beta = 0.2 \). The global maximum is always preserved, since otherwise the curve may be simplified to nothing. Local minima of \( f \) are handled by retaining the persistent maxima of \(-f\).

2.4.3 Curve Approximation via Circular Arcs

Let \( C = \langle p, p_1, \ldots, p_m, q \rangle \) be the curve segment bounded by two local extrema, \( p \) and \( q \). We fit a circular arc with endpoints of \( p \) and \( q \). The center of the circle has to lie on the perpendicular bisector of \( pq \). Let \( O \) be the bisecting point. We first subtract \( O \) from each point in \( C \) so that \( O \) becomes the new origin of the coordinate system. The center of the fitting circle is therefore a point of the form \( c = \lambda u \), where \( u \) is a unit vector perpendicular to the segment \( pq \). To find \( \lambda \), we observe that from Pythagoras’s theorem the radius of the circle is

\[
r = \sqrt{\lambda^2 + \frac{||p-q||^2}{4}}.
\]

We thus look for a value of \( \lambda \) that satisfies the system of \( m \) equations

\[
||p_i - \lambda u|| = r, \quad \text{for} \quad i = 1, \ldots, m ,
\]
approximately. Squaring both sides and rearranging terms yields:

\[ u^T p_i \lambda = \frac{\|p_i\|^2}{2} - \frac{\|p - q\|^2}{8}, \]

and stacking all these equations into a matrix results in the following over-constrained system of linear equations in \( \lambda \):

\[
\begin{pmatrix}
  u^T p_1 \\
  \vdots \\
  u^T p_m 
\end{pmatrix} \lambda = 
\begin{pmatrix}
  \frac{\|p_1\|^2}{2} - \frac{\|p - q\|^2}{8} \\
  \vdots \\
  \frac{\|p_m\|^2}{2} - \frac{\|p - q\|^2}{8} 
\end{pmatrix}.
\]

The least-squares solution can be computed in closed form:

\[ \lambda = \frac{4u^T \sum_{i=1}^m \|p_i\|^2 p_i - \|p - q\|^2 u^T \sum_{i=1}^m p_i}{8 \sum_{i=1}^m (u^T p_i)^2}. \]

### 2.4.4 Results

Sample results are shown in Figure 2.5. Our abstraction process preserves only 1% of the original set of pixels! This reduction comes at the price that not all the abstracted shape resemble the original ones closely. However, the gist of the shapes are by-and-large retained. Simplification is very inexpensive computationally: given binary images of resolution 320 × 240, the shape abstraction process finishes more than 200 images per second on a laptop.
Figure 2.5: Shape simplification. The first and third column show the original shape with color representing the (signed) curvature. The warmer the color, the higher the curvature. The second and the last column show the abstracted shape using typically one percent of the total set of points! Best viewed when enlarged.
Richer applications arise in 2D. In this chapter, we consider labeling an image with multiple tiers. Tiers, one on top of another, enforce a strict vertical order among objects. For example, the sky is above the ground and bottles are placed on top of a table. Ordering may come from physical laws, like gravity, or the typical visual object part arrangement, and is commonly seen in daily life pictures.

Other than the strict order among objects, we often have prior knowledge that is useful for object labeling. One important knowledge is symmetry. Many objects, such as frontal faces, vases, and other man-made structures, are visually symmetric, and we want to enforce the labeling results in the foreground object with a symmetric shape. Another prior is the regularity of the shape of an object. The commonly used measure is the total variation of a boundary curve, which only guarantees that a curve is locally smooth. We define instead the topological smoothness, which bounds the number of extrema of a shape boundary, and is useful for enforcing that a curve is globally smooth and has a given number of local extrema. We explore both symmetry and topological smoothness priors in the multi-tier labeling framework (Figure 3.1).
Figure 3.1: Top left: (five) tiered labeling. Bottom left: symmetric object labeling. Right: the $C$-$D$ boundaries in the top left panel under different topological smoothness constraints. From top to bottom, the curve contains 7, 5, 3, 2, 1 local maxima, respectively.

3.1 Literature Review

Scene labeling assigns to each pixel a semantic label and has been widely studied by Li (1995); Boykov et al. (2001); Ishikawa (2003); Winn and Shotton (2006); Kohli et al. (2009); Liu et al. (2010); Felzenszwalb and Veksler (2010); Strekalovskiy and Cremers (2011). Let $f : \Omega \to \mathcal{L}$ be a labeling function mapping from the image grid $\Omega$ to the label space $\mathcal{L}$. Let $f_p$ be the label of pixel $p$. Labeling can often be modeled as minimizing an energy function in the form of a Markov Random Field (MRF) (see
Li (1995) for a survey):

$$
\min_f \left\{ \sum_{p \in \Omega} D_p(f_p) + \lambda \sum_{(p,q) \in \mathcal{N}} V(f_p, f_q) \right\}
$$

(3.1)

where $D_p(f_p)$ is the data cost of assigning label $f_p$ to pixel $p$, and $V(f_p, f_q)$ is the label inconsistency cost. $\mathcal{N}$ is a neighborhood system: $(p, q) \in \mathcal{N}$ means $p$ and $q$ are neighbors. Finally, $\lambda$ is a regularization parameter that balances the two costs. Clearly, the role of the label inconsistency cost $V$ is to enhance the robustness of the labeling when the data cost $D_p$ is insufficient.

The optimization in Equation (3.1) is known to be NP-hard by Boykov et al. (2001), except when special assumptions and constraints are applied (e.g. Greig et al. (1989); Ishikawa (2003); Kolmogorov and Rother (2007); Liu et al. (2010); Felzenszwalb and Veksler (2010); Strekalovskiy and Cremers (2011)). Of particular interest is the work by Felzenszwalb and Veksler (2010), where an $O(N^{1.5})$ algorithm using dynamic programming is developed for the globally optimal labeling of a three-tiered structure. Here and throughout this chapter, $N$ refers to the number of image pixels and $K$ refers to the number of tiers. Unfortunately, their fast algorithm is designed when $K = 3$ and for the cases when $K > 3$, the standard algorithm with $O(N^K)$ time complexity would be too slow because of its exponential dependency on $K$. Most recently Strekalovskiy and Cremers (2011) extend the computation to multi-tiered labeling using a relaxation of the integer convex programming, but with a greater time complexity in comparison to Felzenszwalb and Veksler (2010) when $K = 3$. Their solution is approximate due to randomized rounding.

Although certain shape priors, e.g. convexity, have been explored in previous work by Felzenszwalb and Veksler (2010); Strekalovskiy and Cremers (2011), to the best of our knowledge both the symmetry prior and the topological smoothness prior have not been studied in the literature of object labeling under the Markov Random
Field formulation. However, the literature on symmetry detection is rich (see Brady and Asada (1984); Davis (1977); Kanade (1981); Ponce (1990); Gool et al. (1996); Lee and Liu (2009); Liu et al. (2004) for examples). Also see Liu et al. (2010) for a recent survey. Three aspects distinguish our work from the literature: First, we do not detect symmetry but focus on finding the detailed shape and location of a known symmetric object. Second, our labeling algorithm is based on a pre-computed cost and is independent of feature matching. Third, we embed symmetry directly into the optimization framework of the Markov Random Field formulation.

The conventional measure of smoothness is to use the total variation to regularize the shape of a tier boundary. However, this measure only encourages that a tier boundary is smooth locally rather than globally. We think that the number of extrema of a boundary curve is a quantity that measures the global smoothness of a curve. Moreover, the number of extrema of a boundary curve can be used as a prior in the tiered labeling and scene label transfer by Liu et al. (2011). This topological measure has been studied in the context of topological persistence and simplification of a triangulated surface by Edelsbrunner et al. (2002, 2003). To the best of our knowledge it has not been considered in MRF optimization or scene label transfer.

Going beyond the current literature, this chapter makes three contributions. First, we develop an $O(KN)$ approximation algorithm to solve the $K$-tier labeling problem. We find it works well in practice and runs in over 100 frames per second on images of VGA resolutions when $K$ is less than 6. When $K = 3$, our solution overlaps with the globally optimal one Felzenszwalb and Veksler (2010) in over 99% of the pixels but runs 1000 times faster. Second, we incorporate symmetry in a binary MRF labeling scheme and demonstrate its usefulness in symmetric object labeling. Moreover, we give an algorithm to compute the optimal labeling of a symmetric object in $O(N^{1.5})$ time. Third, we propose to use the number of extrema to regularize the smoothness of a boundary curve and show that this improves the
accuracy of tiered labeling. We give an efficient $O(MN)$ algorithm to find an optimal tier boundary with at most $M$ local extrema with dynamic programming, and demonstrate improved results in a benchmark data set.

3.2 Tiered Labeling

Tiered labeling divides an image into multiple tiers, each consisting of one or multiple objects. Let $\Omega = [1, \cdots, R] \times [1, \cdots, C]$ be an image grid of $R$ rows and $C$ columns. Given a Directed Acyclic Graph (DAG) $(\mathcal{L}, \prec)$ where $\mathcal{L}$ is a set of labels and $\prec$ is a partial ordering relation defined in $\mathcal{L}$, we have the following definition:

**Definition 5** (Tiered Labeling). A labeling function $f : \Omega \to \mathcal{L}$ is a tiered labeling with respect to $\prec$ if either $f(r, c) = f(r+1, c)$ or $f(r, c) \prec f(r+1, c)$ for each column.
1 ≤ c ≤ C and each row 1 ≤ r ≤ R − 1.

The relation graph can be decomposed to a set of tiers if we run topological sorting on the DAG and group labels that have the same depth from the root. The best algorithm we know of is that of Felzenszwalb and Veksler (2010) which takes $O(N^{1.5})$ time to compute the globally optimal labeling for at most three tiers. The middle tier is allowed to be vertically decomposed to multiple objects while the top and bottom tier contain only a single object. In this section we give an approximate $O(NK)$ labeling algorithm for $K$-tiered labeling using dynamic programming and each tier is allowed to be composed of multiple objects. We first show that 1D tiered labeling can be solved optimally in linear time with respect to the array size. We then solve 2D tiered labeling using 1D tiered labeling as a submodule for cost approximation.

3.2.1 1D Tiered Labeling

We show that 1D tiered labeling can be optimally solved in $O(KN)$ time on a one dimensional array of size $N$ where $K$ is the number of edges in the relation DAG. The problem is to assign each pixel $1 ≤ i ≤ N$ a label $f_i$ so that either $f_i = f_{i+1}$ or $f_i ≺ f_{i+1}$ for $1 ≤ i ≤ N − 1$. The relation $≺$ can be understood as “above” and is imposed a priori. Figure 3.2 illustrates the setting.

We show how to solve the global optimization of Equation (3.1) using dynamic programming. Let $F(i, l)$ be the optimal cost when position $i$ is labeled $l$. Without loss of generality, $l$ takes positive integer values. We then have the following recursive state equation:

$$F(i, l) = \min_{l' ≤ l} \left( F(i - 1, l') + V(l', l) + D_i(l) \right)$$

(3.2)

Dynamic programming computes $F(i, l)$ for each $1 ≤ i ≤ N$ and each $1 ≤ l ≤ K$. 19
Figure 3.3: Decompose a $K$-tiered labeling to a series of $K - 1$ binary labeling. Each can be solved in $O(N)$ time.

The overall time complexity is therefore $O(KN)$. For the boundary conditions, we specify: $F(1, l) = D_1(l)$ for each $1 \leq l \leq K$.

3.2.2 2D Tiered Labeling

We use the optimal 1D tiered labeling to approximate the cost in 2D tiered labeling. Approximations come from two aspects. First, as a pre-processing step, we aggregate the cost of multiple objects within a single tier into a single cost function. Let $D^K$ be the aggregate cost of tier $K$. The set of object labels in tier $K$ is denoted $\mathcal{L}_K$, a subset of $\mathcal{L}$. We define for each pixel $p$:

$$D^K(p) = \min_{l \in \mathcal{L}_K} D_p(l) \quad (3.3)$$

Then, the modified relation graph is reduced to a linear chain. We argue this does not cause serious problems as after the tiered labeling, one can unfold the object labels in each tier. Details are given below.

The second approximation uses a simple strategy to divide the $K$-tier labeling to a series of $K - 1$ binary labeling problems. Each binary labeling problem can be
solved in $O(N)$ time. First, we use the 1D tiered labeling algorithm to compute the cumulative cost $F_c$ for each column $c$. $F_c(i, l)$ is therefore the optimal cost of labeling position $i$ as $l$ at column $c$, and can be computed using Equation (3.2).

Since the modified relation graph is a linear chain, we label each tier as $1, 2, \cdots, K$ from top to bottom. We start from the bottom tier and separate it from the rest of the tiers. This binary labeling problem is equivalent to finding a single path $\{x_c\}_{c=1}^C$ of row indices, one per column. This path separates the bottom tier from the one above it. Here we specify the label inconsistency cost as:

$$V(f_p, f_q) = \begin{cases} 
1 & \text{if } f_p \neq f_q \\
0 & \text{otherwise} 
\end{cases} \quad (3.4)$$

The problem formulation is therefore:

$$\min_{x_1, \cdots, x_C} \left\{ \sum_{c=1}^{C} \mu_c(x_c) + \lambda \sum_{c=1}^{C-1} |x_{c+1} - x_c| \right\} \quad (3.5)$$

where

$$\mu_c(x_c) \triangleq F_c(x_c, K - 1) + \sum_{r=x_c+1}^{R} D^K(r, c) \quad (3.6)$$

stands for the data cost and can be evaluated in $O(1)$ time if an integral image Wikipedia (2012); Viola and Jones (2001) is pre-computed for $D^K$. Let $E_c(x_c)$ be the optimal cost up to column $c$ at pixel $x_c$. The recursive state equation for the global minimization is:

$$E_c(x_c) = \mu_c(x_c) + \min_{1 \leq x_{c-1} \leq R} \{E_{c-1}(x_{c-1}) + \lambda |x_c - x_{c-1}|\} \quad (3.7)$$

Thanks to the generalized distance transform Felzenszwalb and Huttenlocher (2004), Equation (3.7) can be evaluated in $O(R)$ time. Since this dynamic program takes
Figure 3.4: Unfolding object labels by vertical decomposition.

$C$ steps, the overall time complexity is $O(RC)$ or $O(N)$. Once tier $K$ is separated, we proceed to tier $K - 1$ and separate it from the tiers above it in a similar way. Since each time the binary labeling takes $O(N)$ time, the overall time complexity is $O(KN)$. Figure 3.3 illustrates this greedy construction.

We decompose each tier into vertical bands to uncover possible multiple objects (Figure 3.4). This is essentially a 1-dimensional problem because each column within each tier can only have one label. Consider tier $t$ and its label set $\mathcal{L}_t$, and let $C(i,l)$ be the optimal cost of labeling column $i$ as label $l$. The state equation is:

$$C(i,l) = \min \left\{ \lambda + \min_{l' \in \mathcal{L}_t} C(i-1,l'), C(i-1,l) \right\} + D(i,l). \quad (3.8)$$

The time complexity for the above dynamic program is linear in the number of columns, multiplied by the number of labels. Because of the two approximations made above, our algorithm does NOT minimize the exact MRF energy function in Equation (3.1). However, our algorithm guarantees that the solution is a tiered labeling by construction. The advantages of our algorithm lie in its practical efficiency of $O(NK)$ complexity, the ability to label multiple tiers beyond three, and
good performance on par with other methods of greater complexity. For instance, although the algorithm given in Felzenszwalb and Veksler (2010) is globally optimal when $K = 3$, our solution differs from the globally optimal one in less than 1% of the total pixels and runs over 1000 times faster than the $O(N^{1.5}K^2)$ algorithm in Felzenszwalb and Veksler (2010) in our experiments.

3.3 Symmetric Object Labeling

We explore two novel priors within the framework of tiered labeling. The first prior is symmetry. The problem is to label an object embedded in the middle tier of a three tiered structure, assuming that the object is axially symmetric about a horizontal line. We show that the global optimization can be done in $O(N^{1.5})$ time. Note that this is of the same complexity as the algorithm by Felzenszwalb and Veksler (2010).

Our objective can be simplified and reduced to finding a single number $a$, an interval $[I, J] \subset [1, C]$ and a set of positive numbers $\{x_c\}_{c=I}^J$ for the following optimization:

$$
\min_{a,I \leq J, \{x_c\}_{c=I}^J} \left\{ \sum_{c=I}^J \sum_{r=a-x_c}^{a+x_c} D(r,c) + 2\lambda \sum_{c=I}^{J-1} |x_{c+1} - x_c| \right\}.
$$

(3.9)

In this objective, $a$ is the row coordinate of the axis of symmetry. We allow the symmetric shape to be completely contained in the image and no extra tier boundaries are necessary. For ease of discussion, we first fold the cost matrix around a horizontal axis $r = a$ and sum up the corresponding costs (Figure 3.5). Let $F_c(r)$ be the cumulative folded cost summed from the folding axis to row $r$ at column $c$. Here, $F_c(r)$ can be either positive or negative, and $r$ and $c$ are in the coordinate of the folded cost matrix. The objective of Equation (3.9) is equivalent to the following:

$$
\min_{I \leq J, \{x_c\}_{c=I}^J} \left\{ \sum_{c=I}^J F_c(x_c) + \lambda \sum_{c=I}^{J-1} |x_{c+1} - x_c| \right\}.
$$

(3.10)
Similar to the objective in Equation (3.5), the above optimization can be solved by dynamic programming in $O(N)$ time. The difference is that the interval $[I, J]$ is unknown. We use similar ideas in Kadanés’s linear time algorithm for the max-sum sub array Bentley (1986) to transform and simplify the problem. To be more specific, the problem of finding the optimal interval $[I, J]$ is equivalent to that of finding the optimal ending column $J$, where the optimal starting column $I$ for the ending column $J$ can be computed by aggregating the results of column $J - 1$; see Equation (3.11).

$$E_c(x_c) = F_c(x_c) + \min \left\{ \min_{x_{c-1}} \{E_{c-1}(x_{c-1}) + \lambda |x_c - x_{c-1}|\} , 0 \right\} \tag{3.11}$$

Similar to Equation (3.7), we use the generalized distance transform Felzenszwalb and Huttenlocher (2004) as an ingredient to find the optimal $x_{c-1}$ for all possible $x_c$ in a particular column $c$ in $O(R)$ time. We omit the details for brevity. Since we need to enumerate $R$ folding lines, and under each folding line the computation takes $O(R \times C) = O(N)$ time, the overall time complexity is $O(RN)$ or $O(N^{1.5})$ if we assume $R = O(\sqrt{N})$.
Figure 3.6: A cartoon city skyline. This tier boundary induces a large label inconsistency by Equation (3.4) due to sharp transitions. However, the cost is low by the standard of topological smoothness as the skyline contains only two local maxima. The number of local maxima is a topological quantity that is global and tolerates significant sharp transitions. It is also used as a topological prior for tiered labeling in this chapter.

3.4 Topological Smoothness

In the above discussion, we use the label inconsistency cost of Equation (3.4). While this penalty function works generally well in practice, it induces an overly large penalty for very sharp transitions (Figure 3.6). Moreover, the total variation only quantifies the local smoothness of a curve. Many scenes have tier boundaries that are globally smooth in the sense that the tier boundaries contain only one or two local extrema. For instance, in the work of Liu et al. (2010), a scene is decomposed into top, left, right, bottom and middle, and the top and bottom tier boundaries have only one local minimum and one local maximum respectively due to their polygonal representation. We propose to use the number of extrema of a path to quantify its topological smoothness. Our algorithm finds a minimal cost path with exactly $M$
local extrema. It can be modified to find a path with at most $M$ local extrema or $M$ local maxima, within the same asymptotic complexity. Let $F_c(x_c)$ be the total data cost of column $c$ if the path passes through $x_c$. The objective is:

$$\min_{x_1, \ldots, x_C} \sum_{c=1}^{C} F_c(x_c)$$

subject to: path $\{x_c\}_{c=1}^{C}$ has $M$ local extrema

In this constrained optimization, we omit the label inconsistency cost because we expect to use the number of extrema of the path to automatically enhance its regularity. However, including the pairwise smoothness term: $\lambda \sum_{c=1}^{C-1} \|x_{c+1} - x_c\|$ would not increase the computational complexity of our algorithm thanks again to the generalized distance transform. For ease of description, we omit this term in the rest of the discussion. The notion of a local extremum needs to be formally defined first:

**Definition 6 (Local Extrema).** An interval $[I, J]$ is a local maximum of $\{x_c\}_{c=1}^{C}$ if $x_{I-1} < x_I = x_{I+1} = \cdots = x_J > x_{J+1}$. The interval $[I, J]$ is a local minimum of $\{x_c\}_{c=1}^{C}$ if it is a local maximum of $\{-x_c\}_{c=1}^{C}$.

Let $C(r, c, m, \uparrow)$ be the optimal cumulative cost of the path which contains $m$ local extrema before reaching pixel $(r, c)$ through an ascending direction. Similarly, let $C(r, c, m, \downarrow)$ be the optimal cumulative cost of the path that contains $m$ local extrema before reaching pixel $(r, c)$ though a descending direction. We have the following alternating state equations for dynamic programming:

$$C(r, c, m, \uparrow) = F_c(r) + \min \left\{ \min_{r'\geq r} C(r', c-1, m, \uparrow), \min_{r'>r} C(r', c-1, m-1, \downarrow) \right\}$$

$$C(r, c, m, \downarrow) = F_c(r) + \min \left\{ \min_{r'\leq r} C(r', c-1, m, \downarrow), \min_{r'<r} C(r', c-1, m-1, \uparrow) \right\}$$

The state equations utilize the fact that a local maximum is created by an ascending path followed by a descending path and a local minimum is created by a
descending path followed by an ascending path. The symbols ≥ and ≤ include the equality relations so that the path is allowed to move in the same row across adjacent columns.

The boundary conditions need to be posed carefully. First, when \( c = 0 \), we have for each \( 1 \leq r \leq R \) and \( 1 \leq m \leq M \) the following boundary condition:

\[
C(r, 0, m, \uparrow) = C(r, 0, m, \downarrow) = +\infty \tag{3.15}
\]

Second, when \( c \geq 1 \) and \( m = 0 \), we pre-compute the data cost:

\[
B(r, c) = \begin{cases} 
F_c(r) & \text{if } c = 0 \\
B(r, c - 1) + F_c(r) & \text{otherwise}
\end{cases} \tag{3.16}
\]

And the boundary conditions for \( C \) are:

\[
C(r, c, 0, \uparrow) = F_c(r) + \min \left\{ \min_{r' \geq r} C(r', c - 1, 0, \uparrow), \min_{r' > r} B(r, c - 1) \right\} \tag{3.17}
\]

\[
C(r, c, 0, \downarrow) = F_c(r) + \min \left\{ \min_{r' \leq r} C(r', c - 1, 0, \downarrow), \min_{r' < r} B(r, c - 1) \right\} \tag{3.18}
\]

The boundary conditions rule out the case that a constant level curve followed by an ascending or descending path may accidentally induce a local maximum or minimum. The correctness of the dynamic programming then follows easily by induction. The overall time complexity is \( O(MN) \) if a simple book-keeping method is used in evaluating the state equations. Details are omitted for brevity.

### 3.5 Experiments and Demonstrations

We demonstrate applications of multi-tier labeling, symmetric object labeling and labeling under the topological smoothness constraint. For the multi-tier labeling, we first compare our linear time approximation algorithm to the \( O(N^{1.5}) \) algorithm by Felzenszwalb and Veksler (2010) on labeling a three-tiered structure. Although
Figure 3.7: Each row shows 7 out of 60 test images. From top to bottom: Input image, ground truth, generated cost, tiered labeling of Felzenszwalb and Veksler (2010), our result, our result under topological smoothness constraint. Best viewed when enlarged and in the color mode.

Their algorithm is globally optimal in terms of solving Equation (3.1), we show that our greedy construction achieves similar results but runs much faster in practice. We collect 60 images from two benchmark data sets Oliva and Torralba (2001); Quattoni and Torralba (2009) and annotate the ground truth of a three tiered labeling by ourselves. For fair comparison, both algorithms use the same feature costs which are generated as follows: Let $T, O, B$ be the collection of randomly sampled 3D color vectors associated to the top, middle and bottom tier. Let $d(p)$ be the color vector at pixel $p$. We compute the ratio

$$\gamma(p) = \frac{\min_{d \in T} \| d(p) - d \|}{\min_{d \in O \cup B} \| d(p) - d \|}$$

(3.19)
Table 3.1: Average accuracy over 60 images. “+Topology” refers to our tiered labeling under the topological smoothness constraint: the top tier boundary has one local minimum and the bottom tier boundary has one local maximum. Global accuracy is calculated on the whole image. Detail accuracy is counted on the cropped image. See also Figure 3.8 for detailed visual comparison.

<table>
<thead>
<tr>
<th></th>
<th>naive</th>
<th>Felzenszwalb et al.</th>
<th>Ours</th>
<th>+Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Accuracy</td>
<td>0.80</td>
<td>0.96</td>
<td>0.96</td>
<td>0.97</td>
</tr>
<tr>
<td>Detail Accuracy</td>
<td>0.76</td>
<td>0.90</td>
<td>0.91</td>
<td>0.94</td>
</tr>
<tr>
<td>Timing (seconds)</td>
<td>–</td>
<td>10.1</td>
<td>&lt; 0.01</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>Overlap</td>
<td>–</td>
<td>1</td>
<td>&gt; 0.99</td>
<td>–</td>
</tr>
</tbody>
</table>

and we assign the cost associated to $T$ based on the ratio

$$f_T(p) = \begin{cases} 
+1 & \text{if } \gamma(p) > \frac{3}{2} \\
-1 & \text{if } \gamma(p) < \frac{3}{2} \\
+\frac{1}{2} & \text{otherwise} 
\end{cases} \quad (3.20)$$

The rationale behind the cost design is that the closer the feature resembles $T$’s features relative to the background features of $O$ and $B$, the lower the cost. Ambiguous features would receive a cost that is positive. $f_O(p)$ and $f_B(p)$ are generated similarly under cyclic permutation of $T, O, B$.

Figure 3.8: Left to right: the cropped image, result by Felzenszwalb and Veksler (2010), and our labeling with topological constraints (1 local extremum allowed).
Figure 3.7 displays sample results and Table 3.1 shows numerical comparisons. In summary, our algorithm runs 1000 times faster than Felzenszwalb and Veksler (2010) on test images and differs from the optimal solution in less than 1% of all the pixels. Interestingly, our algorithm produces a result that is on par or slightly better than that from Felzenszwalb and Veksler (2010). Accuracy is further improved under the topological smoothness constraint. This improvement is not obvious if tested on the whole image due to the fact that boundaries are thin. To magnify this advantage we crop the boundary regions and test accuracy on this smaller area (Figure 3.8). We achieve 4% improvement over Felzenszwalb and Veksler (2010) in details.

We also show labeling with more than three tiers (Figure 3.9), symmetric object labeling (Figure 3.10) and detailed scene label transfer (Figure 3.11). The number of local maxima or minima is learnt \textit{a priori} and is transferred to the tiered labeling of new test images. We compare the effect of labeling under the total variation and the topological smoothness constraint and show that the topological smoothness constraint keeps the regularity of the labeling and preserves sharp transitions.
Figure 3.9: Right is a 6-tiered scene labeling of the left picture.
Figure 3.10: Symmetric object labeling. In the face picture we deliberately occlude part of the face to showcase that symmetric labeling is useful as a prior for handling partial occlusion.
Figure 3.11: Top: total variation (left) miss the sharp transition captured by topological smoothness (right). Bottom: despite the view change, each image has three local maxima in the tier boundary. This topological quantity is transferred to all the images.
We now turn our attention to 3D applications with topological priors. Our main focus is on recovering the 3D shape of plant root systems. As the primary site of nutrient and water uptake, roots play a critical role in plant growth. Recent research by Moreno-Risueno et al. (2010); Traas and Vernoux (2010) highlights the role of genes in regulating root branching, a key component of overall root architecture. A better understanding of root architecture could lead to the production of plants that sequester larger amounts of carbon dioxide, thus helping to reduce one of the causes of climate change. In addition, improved root systems can aid in food production particularly in marginal soils.

To better understand roots, it is important to be able to compare the complex 3D structure of root systems between plants with different genotypes. In contrast to simple shapes of large volume, plant roots have delicate, fine geometric structures with thin branches; see Figures 4.1 and 4.2 for the plant root imaging system and a sample image. This poses challenges for the image-based 3D reconstruction, which is exacerbated by the inaccuracies caused by unavoidable small refractions and the jittering inherent in the imaging system. Furthermore, there are requirements that
originates from the embedding of the software in a larger work process, which includes the need to have connected 3D reconstructions and software that is efficient and works without user intervention. A sample 3D reconstruction is shown in Figure 4.3 and additional results can be seen in Figure 4.9.

We make three main technical innovations to achieve the detailed 3D reconstruction of plant roots. First, we model the background of each 2D image as a harmonic function, which facilitates the extraction of the silhouette by adaptive thresholding. Second, we formulate the 3D reconstruction step as a compromise between two objectives: satisfying all images and one particular image. The former objective guarantees for a good global approximation and corresponds to the traditional visual hull algorithm. Adding the latter objective, we call this the regularized visual hull algorithm, which reconstructs otherwise lost delicate structures. Third, we present an algorithm inspired by persistent homology developed by Edelsbrunner et al. (2002), in order to guarantee the connectedness of the 3D reconstruction. Our algorithm is efficient and runs fast in practice. For example, given a set of forty images, each consisting of $1,600 \times 1,200$ pixels, we can reconstruct the 3D root structure in seconds on a dual core laptop with only 2 GB memory.
4.1 Literature Review

The problem of reconstructing a 3D shape from 2D images has been studied for decades. The general purpose algorithm referred to as visual hull, or volumetric carving, finds the largest shape consistent with the input silhouettes or color images (See Broadhurst et al. (2001); Culbertson et al. (1999); Kutulakos and Seitz (2000); Laurentini (1994); Matusik et al. (2000, 2002); Seitz and Dyer (1999); Slabaugh et al. (2004); Lazebnik et al. (2007). However, due to its sensitivity to calibration errors, thin features of the shape are likely to be lost. A joint optimization approach by
Guillemaut et al. (2009) has been proposed to cope with the segmentation and calibration errors in the moving camera environment. It is similar to our regularized visual hull but different because it relies on the texture and color information as matching cues, which are not available in our setting. A new imaging system working with coplanar shadowgrams has been introduced in Yamazaki et al. (2007), in which the object and the camera remain still while the light source moves. This reduces the complexity in the calibration step from six degrees of freedom (position and orientation of the camera) to three (position of the light source), and leads to improved reconstruction results. While this method is promising, it cannot be applied in our lab setting in which the opacity of the gel poses challenges to collecting the root shadows.

Complementing the general purpose methods, there has recently been progress using prior knowledge on the shape to be reconstructed. In Faugeras and Keriven (1998); Lhuillier and Quan (2003); Paris et al. (2006), shapes are reconstructed by optimizing objectives that guarantee a continuous and if possible smooth surface. However, these methods assume accurate calibration and cannot deal with jittering
or other movements during the image process. Moreover, these methods are not designed for thin and delicate shapes such as plant roots. Model-based reconstruction of shapes in a restricted class, such as trees, buildings, and human bodies, has also been studied in the past decade (See Martinez et al. (2004); Neubert et al. (2007); Quan et al. (2006); Tan et al. (2007); Xu et al. (2007) for examples). Among these works, image-based tree modeling is the most relevant to our problem. However, this work is geared toward computer graphics applications and aims for trees that look realistic as opposed to being accurate. In particular, fine details are typically not reconstructed but instead artificially generated and added to the reconstruction. In contrast, we consider plant roots for biological studies and therefore aim at a reconstruction that is faithful to the image data and contains as many of the fine details as possible.

To the best of our knowledge, reconstructing delicate shapes and plant roots in particular makes our problem unique. The remainder of the paper describes the novel aspects of our 3D root reconstruction algorithm as well as experimental results that provide evidence for its efficacy.

4.2 Harmonic Background Subtraction

We model an image as a function of intensities, \( J : \Omega \rightarrow [0, 255] \), where \( \Omega \) is the image grid. Assuming it represents a root growing in gel, we define the root as the foreground and the rest of the image as the background. Perhaps the simplest way to separate foreground from background is by splitting the pixels with a single intensity threshold. However, there are drawbacks because the intensity can vary from image to image as well as from one location within an image to another. We therefore propose to work with the normalized intensity, \( I : \Omega \rightarrow [0, 1] \), defined by

\[
I(x, y) = \frac{\sum_{i=0}^{J(x,y)} h[i]}{\sum_{i=0}^{255} h[i]},
\]  

(4.1)
where $h[i]$ is the number of pixels with intensity $i$; compare the first two pictures in Figure 4.5. In the rest of the paper, when we refer to an image, we will mean the normalized intensity function, and we will treat this function as the input to our algorithm.

We find that constructing the foreground with a single threshold can cause significant branch loss, as shown in Figure 4.5, in the middle. We also experiment with hysteresis thresholding by Canny (1986), which works by applying a first threshold to find the main portion of the foreground and then expanding the foreground until a second threshold is reached. This generally improves the quality of the result, as shown in Figure 4.5, second picture from the right. Note, however, that some important fine branches are still missing.

Although the gel medium appears to be non-uniform, we observe that the values vary smoothly over the background and contain no obvious local extrema in the interior. We therefore decide to approximate the background by a harmonic function $B : \Omega \rightarrow [0,1]$. To compute this function, we set $B(x,y) = I(x,y)$ on the boundary and enforce $\Delta B = \frac{\partial^2 B}{\partial x^2} + \frac{\partial^2 B}{\partial y^2} = 0$ in the interior of $\Omega$. In other words, we define the background function by solving the Laplace equation with a Dirichlet boundary condition:

$$B|_{\partial \Omega} = I|_{\partial \Omega}, \quad (4.2)$$
$$\Delta B|_{\Omega-\partial \Omega} = 0, \quad (4.3)$$

where $\partial \Omega$ is the boundary of the domain. Numerically, this partial differential equation with boundary conditions can be solved using the finite element method. The right picture in Figure 4.7 illustrates the method by showing the harmonic background of the root image to its left.

To construct the foreground, we use the difference between the intensity of the image and its background. As we can see in Figure 4.4, the foreground is greatly
enhanced, so that applying hysteresis thresholding results in a qualitatively improved foreground, as shown in Figure 4.5, on the right.

4.3 Regularized Visual Hull

Typically, 3D shapes are reconstructed from foregrounds by the visual hull method. Let $I_k : \Omega_k \to [0, 1]$ be the $k$-th image of a single plant root, for $k = 1, 2, \ldots, N$. For a set $V$ of voxels in 3D, let $\pi_k(V) \subseteq \Omega_k$ be its projection to a set of pixels in the $k$-th image. We write $F_k \subseteq \Omega_k$ for the foreground, noting that $\pi_k^{-1}(F_k)$ is the maximal set of voxels with projection $F_k$. With this notation, we can define the visual hull as
the maximal set of voxels whose projections are contained in all foregrounds:

$$V = \bigcap_{k=1}^{N} \pi_k^{-1}(F_k).$$  \hspace{1cm} (4.4)

Alternatively, we can describe it as the result of an optimization problem. Define the consistency of a voxel $v$ with the $k$-th image as

$$\text{cons}_k(v) = \begin{cases} 
1 & \text{if } v \in \pi_k^{-1}(F_k) \\
-N & \text{otherwise},
\end{cases} \hspace{1cm} (4.5)$$

and its total consistency as $\text{cons}(v) = \sum_{k=1}^{N} \text{cons}_k(v)$. Then the visual hull is the set of voxels that maximizes the total consistency:

$$V = \arg \max_S \sum_{v \in S} \text{cons}(v).$$  \hspace{1cm} (4.6)

It is not difficult to see that the two views of the visual hull are equivalent. To illustrate why the above optimization criterion is not sufficient for our purposes, we use twenty images to reconstruct the root, and assume that most of the images give good quality foreground constructions, as suggested in Figure 4.6. Nevertheless, even tiny distortions can cause inconsistencies between the images such that the
back-projection to 3D is nearly empty. In the end, the visual hull does not match any of the input images. We suggest using one of the twenty images to improve the 3D reconstruction. Our approach is best cast in the optimization framework with an additional regularization term. Given a set of images, one distinguished image $I_j$ in this set, and a regularization parameter $\lambda \geq 0$, the regularized visual hull is the set of voxels, $V_\lambda$, such that

$$V_\lambda = \arg \max_S \left\{ \sum_{v \in S} \text{cons}(v) + \lambda \cdot |\pi_j(S) \cap F_j| \right\},$$

(4.7)

where $|.|$ denotes cardinality.

Note that we propose to use only one image for regularization. The reason is that jittering causes different images to contradict each other, so that using two or more images can result in duplications of the same branch. The limitation to only one distinguished image is not serious since roots are typically thin and cause only a small amount of occlusion. The regularization term may cause more voxels to be added to the solution, but it does not exclude any voxels of the visual hull. It follows that regularized visual hull induces a nested set sequence:

$$V \subseteq V_\lambda \subseteq V_\kappa, \quad \text{for all } \kappa \geq \lambda \geq 0.$$  

(4.8)

We will make use of this observation when we discuss an efficient algorithm for constructing a regularized visual hull.

We now analyze the role of the regularization term and the regularization parameter, $\lambda$. Clearly, regularization encourages the covering of the distinguished foreground, $F_j$. In other words, the new framework introduces an explicit mechanism to use one of the images to guide the 3D reconstruction. If $\lambda$ is small, the distinguished image is not important and the regularized visual hull will barely differ from the visual hull. On the other hand, by choosing $\lambda$ large, we can ensure that each pixel in $F_j$ is covered.
The computation of the regularized visual hull is not difficult. Using the subset relationship expressed in (4.8), we initialize the regularized visual hull to the visual hull: $V_\lambda = V$. Next, we visit each pixel $u$ in $F_j$. If $u$ is not covered, we look for a voxel with maximal consistency measure in the set $\pi_j^{-1}(u)$:

$$v = \arg \max_{v \in \pi_j^{-1}(u)} \text{cons}(v).$$  \hspace{1cm} (4.9)

Note that cons($v$) is negative, else $u$ would already be covered. We then compute the regularized measure, cons($v$) + $\lambda$, and add $v$ to $V_\lambda$ if that measure is positive. Otherwise, we discard $v$.

It is easy to prove the correctness of the above algorithm. The crucial step is to understand the role of equation (4.9). If $v$ is included in $V_\lambda$, no other voxels in the set $\pi_j^{-1}(u)$ will be included, simply because its inclusion would decrease the global consistency measure while contributing nothing to the regularization term.
Hence, the regularized visual hull add the minimal number of voxels to cover the
distinguished image.

4.4 Repairing Connectivity

The regularized visual hull can consist of more than one connected component. However,
for downstream applications, connectedness of the reconstruction is sometimes
required, and we will see that it not difficult to be achieved. We restrict ourselves to
adding voxels to the regularized visual hull, as opposed to removing voxels from it.
When we add a voxel, we prefer those with low inconsistency with the 2D images
and with small distance to the regularized visual hull. For each voxel $v$, we therefore
define

$$\text{incons}(v) = \max\{-\text{cons}(v), 0\}, \quad (4.10)$$
$$\text{dist}(v) = \min_{w \in V_\lambda} \|v - w\|, \quad (4.11)$$

We can now formulate an optimization problem: find a connected set of voxels $U$,
with $V_\lambda \subseteq U$, that minimizes the following two measures in sequence:

1. the maximum distance to $V_\lambda$,

2. the minimum inconsistency with the 2D images.

To be specific, we use the Euclidean distance between the centers of two voxels
to measure their distance, and we say two voxels are neighbors if they share a 2-
dimensional face. A path is then a sequence of voxels in which any two contiguous voxels are neighbors, and $U$ is connected if any two of its voxels have a connecting path within $U$. Similar notions of distance and connectivity are possible and lead to
similar results.

We need some notation to describe an algorithm for this optimization problem.
Let $d \geq 0$ be the smallest threshold such that the set of voxels $S = S_d$ with distance
Algorithm 2 Topology repair

Let $V_\lambda$ and $S$ be given and set $C = S$;
Compute the minimal spanning tree $\mathcal{T}$ of $S$;
for each leaf node $u$ of $\mathcal{T}$ do
    while $u$ is a leaf and $u \notin V_\lambda$ do
        $C \leftarrow C \setminus \{u\}$;
        $u \leftarrow$ the parent of $u$;
    end while
end for

at most $d$ from $V_\lambda$ is connected. We optimize the first criterion by computing $S$ with breadth-first search and limiting $U$ to be a subset of $S$. By definition, $\text{incons}(v) = 0$ if $v \in V_\lambda$, and by construction, $\text{incons}(v) > 0$ if $v \in S - V_\lambda$. Note that $S$ defines a graph in which the voxels are the nodes and pairs of neighboring voxels are the edges. We define the weight of an edge as the larger inconsistency of its two nodes.

Next, we compute the minimum spanning tree of this graph, noting that there are many efficient algorithms described in the literature. In this tree, there is a unique path between any two voxels, namely a minimum cost path that minimizes the maximum weight of its edges. We say $v \in S - V_\lambda$ separates if it lies on such a path connecting two voxels in $V_\lambda$. Finally, the desired solution to our optimization problem is the set $U$ that consists of all voxels in $V_\lambda$ plus all separating voxels of the minimum spanning tree. We compute $U$ by repeatedly removing a leaf node if that node does not belong to $V_\lambda$. The algorithm stops with the desired set $U$. The correctness of the algorithm follows from the fact that for any two nodes in $S$, the minimal cost path that joins them belongs to the minimal spanning tree. After pruning the tree, we are left with all minimal cost paths that connect the components of the regularized visual hull into one component. These paths are aware of the geometry of the root structure because they achieve maximal consistency with the 2D images.

This simple algorithm is sketched in Algorithm 2. Computing the minimal spanning tree takes $O(n \alpha(n))$ time with $\alpha(n)$ the inverse Ackermann function of $n$ and tree pruning takes only $O(n)$ time where $n = |S|$. The overall time complexity is

45
Table 4.1: Comparison of visual hull, the expansion of its results, and the regularized visual hull for four different root systems.

<table>
<thead>
<tr>
<th></th>
<th>(t_1)</th>
<th>(f_1)</th>
<th>(t_2)</th>
<th>(f_2)</th>
<th>(t_3)</th>
<th>(f_3)</th>
<th>(t_4)</th>
<th>(f_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Hull</td>
<td>0.87</td>
<td>0.00</td>
<td>0.93</td>
<td>0.00</td>
<td>0.89</td>
<td>0.00</td>
<td>0.97</td>
<td>0.00</td>
</tr>
<tr>
<td>A Naive Expansion</td>
<td>0.90</td>
<td>0.58</td>
<td>0.98</td>
<td>1.36</td>
<td>0.97</td>
<td>1.06</td>
<td>0.99</td>
<td>0.54</td>
</tr>
<tr>
<td>Regularized Visual Hull</td>
<td>0.92</td>
<td>0.03</td>
<td>0.95</td>
<td>0.01</td>
<td>0.94</td>
<td>0.02</td>
<td>0.98</td>
<td>0.01</td>
</tr>
</tbody>
</table>

therefore \(O(n\alpha(n))\). ¹

4.5 Experiments

For our experimental study of the reconstruction algorithm, we reconstruct forty plant root systems growing in laboratory conditions, each described by forty 2D images taken in a circle around the plant. The root systems are grown in gel containers and vary in shape, size, and complexity. For imaging purpose, these containers are placed on top of a turntable, which is programmed to alternate between a small rotation and a stop, long enough for a single image to be acquired. The consistency of the gel allows for a small motion of the root system during the rotation, which accounts for some of the inaccuracies accumulated during data acquisition.

For camera calibration we use the orthographic projection model, although the more complicated perspective projection model is also applicable. We compare the reconstructions using our regularized visual hull algorithm with those obtained using the conventional visual hull method and with expanded versions of the latter. To quantify the results, we define two measures, called the true positive and the false positive.

¹ Note that the MST algorithm with \(O(n\alpha(n))\) time complexity is too complicated to implement. Instead, we use Kruskal’s algorithm with \(O(n \log(n))\) time complexity. (Cormen et al., 2009, Chapter 23)
positive ratios, denoted as $tp$ and $fp$:

$$tp = \frac{\text{number of covered silhouette pixels}}{\text{total number of silhouette pixels}},$$

$$fp = \frac{\text{number of covered pixels not in silhouettes}}{\text{total number of silhouette pixels}}.$$ 

Note that $tp$ is at most 1, while $fp$ can be larger than 1. We choose this definition to magnify the fact that an improper 3D reconstruction can produce a large number of false positive voxels, in particular in the considered case in which the shape is thin and delicate. Also note that for the visual hull, the false positive ratio is always zero.

To meaningfully compare the regularized with the conventional visual hull algorithm, we expand the reconstruction result of the visual hull result uniformly by a certain radius. The expansion recovers many of the missing voxels, but it also increases the false positive ratio. Note that $fp = 1$ means half of the back projected pixels are incorrect. The comparison of the reconstruction results using the regularized visual hull (RVH), the conventional visual hull (VH), and the expanded results of the visual hull (eVH) is given in Table 4.1. It confirms that the best results are obtained with the regularized visual hull algorithm, as it increases the true positive ratio with only a very modest increase in the false positive ratio. This is nontrivial, because root structures are thin and delicate and therefore increasing $fp$ is much easier than increasing $tp$.

An anecdotal visual comparison is shown in Figure 4.8, where we show the details of the 3D reconstruction by visual hull, the expansion of its result, and our regularized visual hull followed by topology repair. Note that in our experiments, we fixed the parameter $\lambda$ to $6N$. We find that the regularization is crucial in achieving high quality results. In our experiments, the one out of the forty images that was used for improving the reconstruction was chosen randomly. We show a few representative 3D root structures reconstructed with our software in Figure 4.9.
Figure 4.7: The (normalized) intensity of the image, $I$, on the left, and its harmonic background model, $B$, on the right.
Figure 4.8: From left to right: the silhouette, the visual hull, an expansion of the visual hull, and the regularized visual hull after topology repair.
Figure 4.9: Six reconstructed root systems or pairs of root systems, each shown from five different directions.
Conclusions

This thesis has made the following technical contributions to shape simplification, scene labeling, and 3D object reconstruction:

1. Use persistent extrema to simplify while maintaining the gist of a shape.

2. An $O(N)$ algorithm for computing 1D persistence on a curve of $N$ nodes.

3. Use the number of local extrema to bound scene complexity.


5. An $O(N^{1.5})$ algorithm for symmetric object labeling on an image of $N$ pixels.

6. Use harmonic images for background subtraction.

7. Regularized visual hull for preserving the details of an example image.

8. An efficient algorithm for repairing topological connectedness.

Our software for 3D plant root reconstruction has been tested on over 40 real plant roots growing in the lab of the Biology Department at Duke University. It
has also been released as an open source software that can be downloaded at http://benfey-arex2.biology.duke.edu:8002/RootImagingWorkshop-2011-09-15/.

Future works are threefold. First, it is interesting to see if there is a faster algorithm for computing the topological persistence in high dimensional spaces. Second, for tiered labeling, it is useful to explore other topological priors that are relevant in scene reconstruction. Third, for 3D reconstruction, our techniques may be useful for reconstructing other objects containing fine details, such as hair, bones, etc.
Bibliography


Biography

Ying Zheng is a PhD candidate at Duke University. She received her BS in computer science from Fudan University in Shanghai in 2007 and her MS from Duke University in 2010. During her PhD study, she has coauthored over 15 research papers in top conferences and journals in computer vision, multimedia and database systems. She used to work as a research intern in Microsoft Research Asia and Palo Alto Research Center. She was also a software engineer intern at Canesta, Inc. Currently she is working on the next-generation TV at Google YouTube.