10-2014

Component Trees For The Exploration Of Macromolecular Structures In Biology

Lucas Oliveira

Graduate Center, City University of New York

How does access to this work benefit you? Let us know!

Follow this and additional works at: https://academicworks.cuny.edu/gc_etds

Part of the Applied Mathematics Commons, and the Biology Commons

Recommended Citation


This Dissertation is brought to you by CUNY Academic Works. It has been accepted for inclusion in All Dissertations, Theses, and Capstone Projects by an authorized administrator of CUNY Academic Works. For more information, please contact deposit@gc.cuny.edu.
COMPONENT TREES FOR THE EXPLORATION OF MACROMOLECULAR
STRUCTURES IN BIOLOGY

by

LUCAS DE MELO OLIVEIRA

A dissertation submitted to the Graduate Faculty in Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy, The City University of New York

2014
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Date

Dr. Gabor T. Herman, Chair of Examining Committee

Date

Dr. Robert M. Haralick, Executive Officer

Dr. T. Yung Kong

Dr. Katherine St. John

Dr. Jose-Maria Carazo

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK
Abstract

Component Trees for the Exploration of Macromolecular Structures in Biology

by

Lucas de Melo Oliveira

Adviser: Gabor T. Herman

Understanding the three-dimensional structure of a macromolecular complex is essential for understanding its function. A component tree is a topological and geometric image descriptor that captures information regarding the structure of an image based on the connected components determined by different grayness thresholds. This dissertation presents a novel interactive framework for visual exploration of component trees of the density maps of macromolecular complexes, with the purpose of improved understanding of their structure. The interactive exploration of component trees together with a robust simplification methodology provide new insights in the study of macromolecular structures. An underlying mathematical theory is introduced and then is applied to studying digital pictures that represent objects at different resolutions. Illustrations of how component trees, and their simplifications, can help in the exploration of macromolecular structures include (i) identifying differences between two very similar viruses, (ii) showing how differences between the component trees reflect the fact that structures of mutant virus particles have varying sets of constituent proteins, (ii) utilizing component trees for density map segmentation in order to identify substructures within a macromolecular complex, (iv) showing how an appropriate component tree simplification may reveal the secondary structure in a protein, and (v) providing a potential strategy
for docking a high-resolution representation of a substructure into a low-resolution representation of whole structure.
To my family and friends that touched my life in this exciting and arduous journey.

“Praise the name of God forever and ever, for he has all wisdom and power.

He controls the course of world events; he removes kings and sets up other kings.

He gives wisdom to the wise and knowledge to the scholars. He reveals deep and mysterious things and knows what lies hidden in darkness, though he is surrounded by light. I thank and praise you, God of my ancestors...” Daniel 2:20-23
Acknowledgments

I would not have been able to finish my dissertation without guidance from my mentors and committee members, help from friends, and support from my family.

I wish to express my deepest gratitude to my advisor, Dr. Gabor T. Herman, for his patient guidance and the mentoring he provided throughout the years. Thank you for being so accessible and understanding, for sharing your scholarly knowledge, and for inspiring me to be a better researcher. Special thanks are due to my co-mentor Dr. T. Yung Kong for the insightful comments and suggestions regarding this dissertation, especially for his expertise and the mathematical advice that he has given. Also, I wish to thank my committee members Dr. Katharine St. John and Dr. Jose-Maria Carazo for their support and contribution to this work.

Furthermore, I wish to thank Dr. Paul Gottlieb and Dr. Al Katz for their financial support through grant funding and the opportunity to obtain practical experience in biomedical image analysis. I also wish to thank the Computer Science Department of the Graduate Center and the National Science Foundation for the financial support in the early years of my PhD studies.

I wish to thank my colleagues Dr. Ran David, Dr. Wei Chen, and Younes Benkarroum for their contributions and discussions throughout my doctoral studies. Special thanks to my friend Dr. Joanna Klukowska for her constructive comments and warm encouragement. I also thank my invaluable network of supportive, forgiving, and generous friends without whom I could not have survived the process: Joergen Geerds, Uli Futschik, Thiago Penteado, Taina Penteado, Hercules Andrade, Raphael Quintela, Thais Oliveira, Felipe Santos, Jair Fernandes, William Miranda, Lina Garcia, Andre Pitanga, and Joe Driscoll.

I wish to show my greatest appreciation to Drs. Bruno Carvalho and Simone Nunes. Their friendship, guidance, and encouragement in my graduate studies were fundamental to reach this stage in academic life.
I owe my deepest gratitude to my wife Justina Oliveira. Thanks for the extensive and careful review of this text. Your presence in my life makes this journey more enjoyable. I am also most grateful to my parents, siblings, and family in Brazil for the support and love throughout my life. Minha eterna gratidão aos meus pais Pedro Ivo e Ana Maria pelo apoio incondicional na minha longa jornada educacional. O que eu aprendi com voces é o que eu tenho de mais precioso. Aos meus irmãos Lucio, Liana, Leandro e Bartolomeu um agradecimento especial pelo suporte e carinho durante meu doutorado.
# Contents

1 Introduction 1

2 Background 8

2.1 High and Low-Resolution Structural Information 8
   2.1.1 Low-Resolution Structural Information 10
   2.1.2 High-Resolution Structural Information 12
   2.1.3 Generation of Density Maps from Atomic Structures 13

2.2 Docking High-Resolution Structures into a Low-Resolution Density Map 17
   2.2.1 Rigid-body Docking 20
   2.2.2 Flexible Docking 23
   2.2.3 Multiple Docking 24

2.3 The Embedded Tree Problem and Its Algorithms 25

3 Digital Pictures and their Component Trees 29

3.1 Tree Representation of Digital Picture Embeddings 30
   3.1.1 Embedded Digital Spaces and Pictures 30
   3.1.2 Connectedly Embedded Digital Pictures 38
   3.1.3 The Component Tree of a Digital Picture 43
      3.1.3.1 Component Tree of a 1-Dimensional Digital Picture 44
3.1.3.2 Component Tree of a 2-Dimensional Digital Picture .......................... 50
3.1.4 Tree Embeddings .................................................. 54
3.1.5 Illustration of Potential Applicability in Structural Biology .................. 58
3.2 Component Trees of Multidimensional Images ................................. 62
  3.2.1 Foreground Component Tree Structure .................................. 63
  3.2.2 A Robust $(\lambda,k)$-Simplification of an FCTS ......................... 68
  3.2.3 Essential Isomorphism and Level-Preserving Theorem .................... 70
  3.2.4 Pruning by Removing Branches of Length $\leq \lambda$ ...................... 77
    3.2.4.1 Specification of Simplification Step 2 ................................. 77
    3.2.4.2 An Easily Visualized Characterization of the Output of Simplifica-
    tion Step 2 ........................................................................ 78
    3.2.4.3 Linear-Time Implementation of Simplification Step 2 ............... 82
  3.2.5 Elimination of Internal Edges of Length $\leq \lambda$ from $\hat{\mathcal{S}}_{\text{crit}}$ .. 84
    3.2.5.1 Specification of Simplification Step 3 .................................. 84
    3.2.5.2 Implementation of Simplification Step 3 .............................. 86
  3.2.6 A Potential Application of FCTSs ...................................... 88

4 Using Component Trees to Explore Biological Structures .......................... 92
  4.1 MataExplor: An Interactive Tool for Component Tree Exploration .......... 94
  4.2 Examples of Macromolecular Exploration Using Component Trees ...... 100
    4.2.1 Interactive Component Tree Exploration .................................. 100
    4.2.2 Component Tree Exploration by Threshold Level ....................... 102
    4.2.3 Automatic Component Tree Exploration .................................. 105
  4.3 Case Study 1: Using Component Trees to Identify Proteins in the
    Procapsid of the Bacteriophage $\phi 6$ ...................................... 106
4.4 Case Study 2: Using Component Trees to Investigate the Structure of a Protein .................................................. 115

5 Investigation of the Use of Component Trees for Macromolecular Docking ............................................. 125
   5.1 A Tentative Docking Methodology Based on Component Trees .................................................. 126
   5.2 A Simple Example of Docking Using Component Trees .............................................................. 131
   5.3 A Proposed Evaluation Methodology ......................................................................................... 136

6 Conclusion ........................................................................................................................................... 143
   6.1 Contributions ............................................................................................................................... 143
   6.2 Suggestion for Future Work ....................................................................................................... 145

Appendix A Some Properties of Simplification Steps 2 and 3, and a Proof of the Correctness of Algorithm 1 146
   A.1 Properties of Simplification Step 2 ........................................................................................... 146
   A.2 Properties of Simplification Step 3 ........................................................................................... 153
   A.3 Justification of Algorithm 1 ...................................................................................................... 155

Appendix B A Constructive Proof of Theorem 4 .................................................................................. 158
   B.1 Step 1 of the Proof of the Lemma B.1 ..................................................................................... 160
   B.2 Some Useful Observations ........................................................................................................ 161
   B.3 Step 2 of the Proof of the Lemma B.1 ..................................................................................... 164
   B.4 Step 3 of the Proof of the Lemma B.1 ..................................................................................... 169

Appendix C Justification of Assertions L, M, N, and O in Step 3 of the Proof of the Lemma B.1 173
   C.1 Proof of Assertion L ................................................................................................................. 174
   C.2 Proof of Assertion M ................................................................................................................. 175
   C.3 Proof of Assertion N ................................................................................................................. 176
List of Tables

4.1 The Density Maps of Four Recombinant Procapsids of Bacteriophage φ6. ......................................................... 109
List of Figures

2.1 Series of Resolutions for GroEL + GroES. .......................... 9
2.2 Density Map for GroEL. .................................................. 12
2.3 The Ribbon Diagram and PDB File for the Atomic Model of GroEL
    at 2.8 Å. ................................................................. 14
2.4 A Three Step Docking Pipeline. ................................. 18

3.1 A Digital Picture \((V, \pi, f)\). ........................................... 33
3.2 Embedded Digital Picture. ............................................. 35
3.3 Connectedly Embedded Digital Picture. ......................... 37
3.4 A Simple 1-Dimensional Digital Picture. ....................... 44
3.5 Detailed and Simplified Representations of a Component Tree. ... 45
3.6 A Simple 2-Dimensional Digital Picture. ....................... 50
3.7 Components of the Digital Picture of Figure 3.6(c). ............ 51
3.8 Component Trees of Digital Pictures. .......................... 53
3.9 Embedding of Component Trees. ................................... 54
3.10 Illustration of Potential Applicability of Embedded Digital Pictures. 58
3.11 Illustration of Potential Applicability of Embedded Component Trees. 59
3.12 A Rooted Tree in which the Critical Nodes have been Circed. ... 64
3.13 The Tree of the FCTS that is Defined in Example 1. ........... 65
3.14 The FCTS of a Digital Picture \((V, \pi, f)\). ......................... 67
3.15 Critical Nodes of FCTS ............................................. 69
### Index

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.16</td>
<td>The Effect of Pruning an FCTS by Removing Nodes of Size $\leq k$</td>
<td>71</td>
</tr>
<tr>
<td>3.17</td>
<td>Example of $K(V, \pi, f)$ and $A(V, \pi, f)$</td>
<td>72</td>
</tr>
<tr>
<td>3.18</td>
<td>The Effect of Pruning an FCTS by Removing Branches of Length $\lambda$</td>
<td>79</td>
</tr>
<tr>
<td>3.19</td>
<td>The Effect of Eliminating Internal Edges of Length $\leq \lambda$ from an FCTS</td>
<td>86</td>
</tr>
<tr>
<td>3.20</td>
<td>Two Different Versions of Adenovirus</td>
<td>89</td>
</tr>
<tr>
<td>3.21</td>
<td>$(\lambda, k)$-simplifications of FCTSs of Wildtype and Mutant Adenoviruses</td>
<td>90</td>
</tr>
<tr>
<td>4.1</td>
<td>MataExplor Flowchart</td>
<td>95</td>
</tr>
<tr>
<td>4.2</td>
<td>MataExplor Screenshot</td>
<td>96</td>
</tr>
<tr>
<td>4.3</td>
<td>Zooming a Component Tree Using MataExplor</td>
<td>99</td>
</tr>
<tr>
<td>4.4</td>
<td>Microtubule Binding Patterns of Dimeric Kinesins</td>
<td>101</td>
</tr>
<tr>
<td>4.5</td>
<td>Interactive Component Tree Exploration</td>
<td>102</td>
</tr>
<tr>
<td>4.6</td>
<td>Component Tree Exploration by Threshold Level</td>
<td>104</td>
</tr>
<tr>
<td>4.7</td>
<td>Automatic Component Tree Exploration</td>
<td>105</td>
</tr>
<tr>
<td>4.8</td>
<td>Approximate Positions of the Proteins in a Wild-Type $\phi 6$ PC</td>
<td>108</td>
</tr>
<tr>
<td>4.9</td>
<td>Central Slices and Component Trees of Mutant $\phi 6$ PCs</td>
<td>110</td>
</tr>
<tr>
<td>4.10</td>
<td>$(2,800)$-Simplified Component Tree of the PC14 Component Tree</td>
<td>111</td>
</tr>
<tr>
<td>4.11</td>
<td>$(2,800)$-Simplified Component Tree for the PC124 Component Tree</td>
<td>112</td>
</tr>
<tr>
<td>4.12</td>
<td>Identifying Proteins in PC147 Component Trees</td>
<td>113</td>
</tr>
<tr>
<td>4.13</td>
<td>Identifying Proteins in PC1247 Component Trees</td>
<td>114</td>
</tr>
<tr>
<td>4.14</td>
<td>Four Protein Hierarchical Levels</td>
<td>117</td>
</tr>
<tr>
<td>4.15</td>
<td>Using Component Trees to Explore Protein Secondary Structure</td>
<td>119</td>
</tr>
<tr>
<td>4.16</td>
<td>Secondary Structure Elements for the Bacteriorhodopsin</td>
<td>121</td>
</tr>
<tr>
<td>4.17</td>
<td>Secondary Structure Elements for the Triose Phosphate Isomerase</td>
<td>124</td>
</tr>
<tr>
<td>5.1</td>
<td>A Tree Docking Methodology Pipeline</td>
<td>128</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.2</td>
<td>An Illustrative Example of the Proposed Docking Methodology</td>
<td>132</td>
</tr>
<tr>
<td>5.3</td>
<td>Using Component Trees for Macromolecular Docking</td>
<td>135</td>
</tr>
<tr>
<td>5.4</td>
<td>Figures of Merit: RMSD and OS</td>
<td>138</td>
</tr>
<tr>
<td>5.5</td>
<td>Demonstration of the Proposed Experiments with Synthetic Data</td>
<td>140</td>
</tr>
<tr>
<td>5.6</td>
<td>Experimental Data sets</td>
<td>142</td>
</tr>
</tbody>
</table>
Nomenclature

Å  Angstrom is a unit of length equal to $1 \times 10^{-10}$ m

$\rho$  The intensity for a voxel in a density map computed using Gaussian spheres, see Eq. (2.1)

$\gamma$  Half the desired resolution distance, see Eqs. (2.1) and (2.2)

$m_i$  Atomic weight (or mass in Daltons) for an atom $i$, see Eq. (2.1)

$F_e$  Electron scattering factor of a given atom in spacial frequence, see Eqs. (2.6) and (2.7)

$\rho_{easf}$  The intensity for a voxel in a density map computed using electron scattering factor, see Eq. (2.8)

$\tilde{\rho}_{easf}(r)$  Low-pass Electron Atomic Scattering Factors (LEASF), see Eq. (2.9)

$I_t$  Target density map

$I_m$  Model density map

$\mathbb{R}$  Set of all real numbers

$CC$  Cross-correlation between two density maps, see Eq. (2.10)

$CC_L$  Local cross-correlation, see Eq. (2.12)

$T = (N, E)$  Rooted tree, where $N$ is a finite set of nodes and $E$ is a set of edges
$l_T : N \rightarrow L$ Labeling function mapping nodes to the set of label $L$

$L$ Set of all label nodes

$C \prec_T D$ $C$ is a proper ancestor of $D$ in $T$

$C \preceq_T D$ $C$ is an ancestor of $D$ in $T$

$C \succ_T D$ $C$ is a proper descendant of $D$ in $T$

$C \succeq_T D$ $C$ is a descendant of $D$ in $T$

$\varphi_i$ Isomorphism map

$\varphi_e$ Embeddable map

$V$ Nonempty finite set

$\pi$ Symmetric irreflexive binary relation on $V$

$\langle d^{(0)}, \ldots, d^{(K)} \rangle$ Sequence of elements from $d^{(0)}$ to $d^{(K)}$

$(V, \pi)$ A digital space

$(V, \pi, f)$ A digital picture over the digital space $(V, \pi)$

$f$ A function that maps $V$ into $\mathbb{R}$

$\xi : V_1 \rightarrow V_2$ Bijective correspondence that defines isomorphic digital pictures

$V_{f,t}$ $t$-superlevel set of $(V, \pi, f)$

$C_{(V, \pi, f)}(c)$ $\pi$-component of $(V, \pi, f)$ defined by $c \in V$

$\mathcal{C}_{(V, \pi, f)}$ Set of components of $(V, \pi, f)$

$V - V'$ Set theoretical difference between $V$ and $V'$

$T_{(V, \pi, f)}$ Component tree for the digital picture $(V, \pi, f)$
\( \phi \quad \text{Natural component embedding associated} \)

**CCD(}C) Closest critical descendant for the node \( C \)

**Nodes(}T) Set of all nodes of \( T \)

**root(}T) The root of \( T \)

**Leaves(}T) set of all leaves of \( T \)

**Children\(_T\)(C) Set of all the children of \( C \) in \( T \)

**parent\(_T\)(C) Parent of \( C \) in \( T \)

**FCTS foreground component tree structure

\( (T, \ell) \quad \text{Pair that defines a FCTS} \)

**\( \mathcal{E} \quad \text{Collection of nonempty } \pi\text{-connected sets of spels} \)

\( \ell \quad \text{Real-valued function on } \mathcal{E} \text{ (level of an element of } \mathcal{E} \text{)} \)

**\( \mathcal{F} \quad \text{Rooted tree } T \text{ for the FCTS } (T, \ell) \)

**FCTS\(_{(V,\pi,f)}\) The FCTS of the digital picture \( (V, \pi, f) \)

**node\(_T\)(c) The smallest node of **Nodes\(_T\) that contains \( c \).

\( (V, \pi, f) \mathcal{F} \quad \text{The digital picture of the FCTS } \mathcal{F} \)

**Crit(T) Set of all critical nodes of \( T \)

**LCN(T) Lowest critical node of \( T \)

**\( T^{\text{crit}} \) Rooted tree whose set of nodes is **Crit\(_T\) \( \cup \{ \text{root}(T) \} \)

**\( \theta \quad \text{Map that defines an essential isomorphism} \)

**\( K_{(V,\pi,f)} \quad \text{Smallest element in } \text{Leaves}\(_{T(V,\pi,f)}\)\)
\( \Lambda_{(V, \pi, f)} \) Minimum difference between two critical nodes in \( T_{(V, \pi, f)} \) such that the first is a proper ancestor of the second

\( C \downarrow_T \) Set of all ancestors of \( C \) in \( T \)

\( C \downarrow_T \) Set of all proper ancestors of \( C \) in \( T \)

\( C \uparrow_T \) Set of all descendants of \( C \) in \( T \)

\( C \uparrow_T \) Set of all proper descendants of \( C \) in \( T \)

\( \wedge_T S \) Closest common ancestor of the set of nodes \( S \)

\( \text{leaf}[1], \ldots, \text{leaf}[n] \) An \( \ell_{in} \)-increasing enumeration of \( \text{Leaves}(\mathcal{F}) \)

\( \text{leaf}[n] \) The \( n \)-th leaf in the \( \ell \)-increasing enumeration of \( \text{Leaves}(\mathcal{F}) \)

\( T[C] \) The subtree of \( T \) that is rooted at \( C \)

\( \mathcal{U}^\lambda(\mathcal{F}) \) Set of all \( C \in T \) for the FCTS \( \mathcal{F} = (T, \ell) \) such that \( \text{depth}_\mathcal{F}(C) > \lambda \)

\( \mathcal{V}^\lambda(\mathcal{F}) \) Set of all \( C \in T \) of the rooted tree \( T \) of \( \mathcal{F} = (T, \ell) \) such that \( C \notin \mathcal{U}^\lambda \) but \( C \downarrow_T \subseteq \mathcal{U}^\lambda(\mathcal{F}) \)

\( \text{lastLeaf}_\sigma(C, T) \) The leaf of the subtree \( T[C] \) that occurs later in a \( \ell \)-increasing enumeration \( \sigma \) than all other leaves of \( T[C] \)

\( \text{Path}_\sigma(C, T) \) The set of all \( D \in \text{Nodes}(T) \) such that \( C \preceq_T D \preceq \text{lastLeaf}_\sigma(C, T) \)

\( \mathcal{D}(\mathcal{F}) \) Set of real numbers determined by the difference between two non-leaf critical nodes in \( \mathcal{F} \) such that the second is a proper ancestor of the first.

\( \text{pred}_\mathcal{F}(\lambda) \) Maximum element of \( \mathcal{D}(\mathcal{F}) \) that is smaller then \( \lambda \)

\( h(\mathcal{F}_{in}, \lambda) \) The length \( l \geq 1 \) of the longest chain of nodes \( C \succ_{T_{in}} \cdots \succ_{T_{in}} C_l \) in \( \text{Crit}(\mathcal{F}_{in}) \)
\( \omega_{T', T} \) The minimal component inconsistency over all root-to-root tree embeddings of \( T' \) into \( T \).

\( \sigma_{T', T} \) The minimal value of \( \omega_{C'_0, T} \) over all nodes \( C' \) in \( T' \).

**RMSD** Compute the RMSD using the coordinates for all the elements of a digital picture.

**RMSD\_h** Compute the RMSD using the position of the atomos in an atomic model.

\( C_{(V, \pi, f)}(s, \tau) \) The the set of all \( s' \in V \) for which there exists a \( \pi \)-path \( s_0, \ldots, s_l \) such that \( s_0 = s, s_l = s', \) and \( f(s_i) \geq \tau \) for \( 0 \leq i \leq l \).

\( c \in_{(V, \pi, f) \geq \tau} d \) The spels \( c, d \in V \) such that \( d \in C_{(V, \pi, f)}(c, \tau) \).

\( \mathcal{M}(C_{(V, \pi, f)}(v)) \) The set of leaves of the subtree \( T \) rooted in \( C_{(V, \pi, f)}(v) \).

\( \mathcal{L}_T C \) The set of leaves of the subtree \( T \) rooted in \( C \).

**depth** \( \mathcal{S}(C) \) The maximum difference between \( \ell(D) - \ell(C) \) where \( T[\mathcal{C}] \) is a subtree of \( \mathcal{S} \) and \( D \in \text{Leaves}(T[\mathcal{C}]) \).
Chapter 1

Introduction

Three-dimensional (3D) structural studies of biological matter, from proteins to whole cells, are of a great importance for fully understanding the function of macromolecular complexes and organelles within cells. Over the years, several experimental techniques have been proposed to image biological specimens and produce useful information for understanding their function and evolution. These techniques reveal different levels of a macromolecular structure: low-resolution techniques such as cryo-Electron Microscopy (cryo-EM) reveal the overall molecular shape and techniques such as X-ray crystallography provide an atomic structural model of subunits for macromolecular assemblies.

An atomic model reveals important structural details of a subunit and is useful for assigning functional properties to 3D reconstructions of macromolecular assemblies. On the other hand, atomic structures are often obtained in functionally undefined states and in some cases structures cannot be imaged at high-resolution due to their particular crystal organization or size. In these circumstances, cryo-EM can be used as a complementary method, producing a structural model of the sample in its native state. Even with a limited resolution, cryo-EM captures various function states and is fundamental to the structure determination of an assembly.
Due to the large number of macromolecular assemblies and atomic structural subunits imaged in the last few decades, biological databases were developed to organize and store information from biological experiments. These public repositories, both for electron microscopy information [1, 2] and atomic structural information [3, 4, 5], have provided free access for a diverse set of biological specimens. Using these databases, the scientific community can study biological structures in order to understand the organization and evolution of macromolecular assemblies.

Combining an atomic model of a part of a macromolecular assembly with low-resolution information of the whole macromolecular assembly gives a more detailed picture of the intact assembly [6, 7]. Baker and Johnson [8] asserted that this combination could yield a very useful pseudo-atomic precision model for the study of macromolecular assemblies and can trigger new insights for structural biology.

However, in order to create these pseudo-atomic models, a question needs to be answered: Where is the position of the atomic model in the whole macromolecular assembly? This query is relevant as the atomic model and the macromolecular assembly image are usually produced independently by different techniques and with different levels of detail. As will be discussed in Chapter 5, the component tree can be used to help answer this question.

**Digital Representation of Macromolecular Information**

A basic assumption in image processing is that the essential information of a continuous object can be converted into a discrete image. In reality, this assumption is based on the fact that computers cannot handle (i.e., store and process) continuous functions. Fortunately, if a continuous function has certain properties, such as if it is band-limited, a continuous function can be represented by a set of discrete samples.

A function is said to be *band-limited* if the frequency components of the function are zero above a certain finite range (or band). A significant fact is that a
band-limited function can be fully reconstructed from its samples. According to
Nyquist-Shannon’s sampling theorem [9], a band-limited continuous function can
be represented by a set of discrete samples of the original function taken at regular
intervals (or step-size). The Nyquist-Shannon sampling theorem says that a perfect
reconstruction of a continuous function is possible when the sampling frequency
(i.e., the number of samples per unit of space) is greater than twice the maximum
nonzero frequency of the signal being sampled [9].

The level of detail described by a sampled function is indicated by the size of
the radius of a ball centered at the origin in the reciprocal space that contains the
3D domain within which Fourier components contribute significantly to the density
map [10]. This spatial frequency radius $R$ is called the bandlimit or resolution
limit. The bandlimit defines the maximum frequency used in the function (i.e., no
information is used outside of the spherical region with radius $R$). According to
the sampling theorem, to image all details of a function that has a bandlimit $B$, a
sampling frequency with a step size equal to $1/2B$ is required.

In this document the term resolution is used to refer to the size of the bandlimit.
A low-resolution image (or density map) for example, is produced with a relatively
smaller bandlimit than a high-resolution image (or density map). This reciprocal
space value is measured in Å$^{-1}$s (Å stands for Angstrom and 1 Å is equal to $1.0 \times$
$10^{-10}$ meters).

A complete discussion of the definition of the resolution of an image is beyond
the scope of this dissertation. It is important to mention that in the 3D electron mi-
croscopic literature a real space value is also used to represent the term resolution.
This value is usually associated with the "theoretical resolution" and is measured
in Ås (for more detail see the discussion of "Resolution Assessment" in [11]). To
avoid confusion, we use the term resolution distance when referring to the theore-
tical resolution.
In this dissertation a density map or image is considered to be a 3D array of real numbers produced by an imaging device or a computer procedure. The grid (or lattice) is the set of points in the physical space at which a physical quantity is measured or estimated. The sampling frequency used in this estimation defines the grid step-size (or voxel size) for the points in the grid (i.e., the distance between adjacent data elements along the same axis) \[12\]. A smaller grid step-size allows the capturing of more details (corresponding to a higher resolution) in the image. 

To illustrate the resolution concept used in our work, consider the case of imaging a typical protein that has a diameter around 200 Å. If we desire to produce density maps with atomic detail, we need to choose an appropriate sampling frequency. Sorzano et al. \[13\] state that, for visualizing the atomic structure, a sampling frequency of 12 Å\(^{-1}\) (and, consequently a sample step-size of 1/12 Å) should be used. Using 12 Å\(^{-1}\) as sampling frequency, the high-resolution density map for a typical protein should use at least \(2400 \times 2400 \times 2400\) (\(2400 = 200 \times 12\)) grid points.

**Visual Exploration of Macromolecular Information**

Computer-based visualization is extensively used in biological studies to help understanding and communicating data, to generate ideas, and to interpret the architecture and operation of biological machinery \[14, 15\]. The visual exploration of density maps for macromolecular assemblies provides structural information that can produce new insights in the study of biological specimens.

There are several types of software available to visually explore biological data. The first visualization tools for biological visualization were stand-alone programs used by a few experts in the field. Today, with hardware and network development, the amount of available software has increased significantly and in some instances it includes a complex package of tools for a large variety of applications.

In a recent review by O’Donoghue et al. \[16\] on the present and future of
biological visualization tools, the authors discuss some user-interface and computational challenges involved in the representation of biological information such as usability, multiscale navigation, and innovative representations to present macromolecular information. Another challenge in the development of visualization tools addressed by [16] is the absence of a standard notation in the biological community. The visualization tools were first developed by labs or research communities that were in need of tools to explore their data in a very specific manner. Therefore, this non-centralized growth in the biological software community led to a non-uniform standard for digital image representation, notation, and definition among other issues. Fortunately, there has been some effort in the scientific community to create a common convention for basic definitions, representations, and interpretations of 3D EM, as can be seen in [12, 17, 18].

It is important to mention that visualization tools can depict macromolecular structures with a large range of detail, from atoms and bonds to large macromolecular assemblies. The levels of detail can be based on the macromolecular assemblies hierarchies of structure: atoms and bonds, residues, helices and sheets, domain, macromolecules, and complexes [15]. Most of the visualization tools depict biological information using standard representations such as ribbon diagrams, ball-and-sticks, and surface renderings. (For a list of resources for visualizing macromolecules based on what type of biological question these methods and tools can help answer, see [14].)

As mentioned earlier, an important question in the study of biomolecular structures is how to find the position in a low-resolution density map (obtained from cryo-EM) of a high-resolution atomic structure (obtained from X-ray crystallography or NMR spectroscopy). Several methodologies have been proposed to find the position of (i.e., to dock or fit) an atomic model into a macromolecular assembly (a short review for these methodologies will be presented in Section 2.2). Roughly
speaking, a fitting (or docking) methodology computes the “position” of the model density map (the high-resolution density map) in the target density map (the low-resolution density map) using a quality-of-fit (QOF) measure. Docking is done by either a manual or a computational procedure that aims to maximize the QOF measure used.

The core task of a docking procedure is to find the spatial relationship between a low-resolution density map of a complete biological specimen and atomic models of one or more of its subunits. The large computation time, the variety of possible structural conformations of the subunits, and the lack of detail in low-resolution maps are some of the challenges that docking methodologies need to overcome.

This dissertation proposes a novel interactive framework for the visual exploration of macromolecular information, yielding efficacious new tools for the study of biological data. The proposed framework is based on component trees, which are data structures that capture topological and geometric information based on the connected components determined by different graynesses in an image. As we will discuss later, a component tree is a topologically invariant descriptor that can be very useful for dealing with biological structures. In addition, we discuss ideas regarding implementation for graph-algorithmic approaches that use trees computed from high- and low-resolution density maps to produce a list of possible or even exact locations for the high-resolution atomic structures in a low resolution target density map.

The two types of biological information investigated in this dissertation as well as a short review of macromolecular docking methodologies and tree embedding problems are presented in Chapter 2. Chapter 3 reports on two of our previously-published works that involve component trees and relevant experiments with biological data. The first paper [19] describes a mathematical theory of the intuitive concept of embedded pictures and connectedly embedded pictures, while the sec-
ond [20] describes a provably robust component tree simplification. Next, Chapter 4 describes the proposed interactive framework and presents several examples from our previously-published works [21, 22, 23] regarding the use of component trees for exploring biological information. In Chapter 5, a potential methodology that uses component trees for macromolecular docking is described. Chapter 6 briefly discusses our contribution and the suggested future work. Lastly, Appendices A, B, and C provide some mathematical details and the proofs for some lemmas, theorems, and algorithms that have been left out in the main text in order to improve its readability.
Chapter 2

Background

In this chapter we present the background necessary for developing our work. In Section 2.1, we cover in detail the two main inputs for the visualization methodology proposed in this dissertation: an atomic model of part of a macromolecular assembly and a low-resolution density maps for the whole macromolecular assembly. In the following section, Section 2.2, we discuss the problem of docking a high-resolution density map obtained from an atomic model into a low-resolution density maps. In the last section, we present a brief review for the embedded tree algorithm and its applications.

2.1 High and Low-Resolution Structural Information

Three-dimensional (3D) structural studies of biological matter, from proteins to whole cells, are of great importance for fully understanding the function of macromolecular complexes and organelles within cells. The 3D structure of a cellular component is tightly related to its function within a cell, and the knowledge of both structure and function is necessary, for instance, to design drugs whose targets are particular proteins.

The quality of a 3D biological structure can be expressed in terms of resolu-
Surface rendering from density maps, using four different resolution distances for GroEL + GroES: (a) 4 Å, (b) 8 Å, (c) 16 Å, and (d) 32 Å. The details are smeared away as the resolution becomes lower (that is, the resolution distance increases). The displays were computed by the program MolMap (Chimera visualization tools [25]) using the atomic model with PDB ID 1AON [24].

We can think of resolution as the amount of detail provided by an image. A high-resolution density map for example, can provide a sharp and detailed structure description, while a low-resolution density map gives a smooth and simple structure description. Figure 2.1 shows the macromolecule GroEL + GroES [24] at four different resolution distances.

Several complementary techniques have been developed for determining the 3D structure of biological specimens. Each one of these techniques reveals a different level of a macromolecular structure. X-ray crystallography is a method of determining the arrangement of atoms within a crystal, in which a beam of X-rays strikes a crystal and causes the beam of light to spread into many specific directions. X-ray crystallography produces high-resolution information and an atomic model of an imaged biological specimen. The atomic model is useful for revealing important structural details of macromolecular subunits and assigning functional properties to macromolecular assemblies.

X-ray crystallography produces high-resolution information with resolution at a near atomic level. However, a large number of macromolecules diffract poorly or
cannot be crystallized. In these cases, Nuclear Magnetic Resonance (NMR) or 3D Transmission Electron Microscopy (TEM) techniques can be chosen to investigate the structure depending on the macromolecule weight.

The NMR technique produces high-resolution information by exploring the magnetic properties of certain atomic nuclei to determine physical and chemical properties of atoms or the molecules in which they are contained. The TEM is a microscopy technique whereby a beam of electrons is transmitted through an ultra-thin specimen, interacting with the specimen as it passes through. In contrast with NMR and X-ray techniques, TEM usually produces low-resolution structural information of the specimen studied.

Cryo-electron microscopy (cryo-EM), or electron cryomicroscopy, is a form of TEM where the sample is studied at cryogenic temperatures (generally liquid nitrogen temperatures) allowing the preservation of the native environment of the specimen. Cryo-EM has proven indispensable for producing reliable images of intact biological structures [26]. Single particle cryo-EM reconstruction, which addresses the problem of determining the structure of macromolecular complexes from projection images, has become a standard method of analyzing structures [11].

### 2.1.1 Low-Resolution Structural Information

In order to compute a density map for a biological specimen, it is necessary to compute a sequence of values that represent the molecular density for this specimen. In contrast with the X-ray crystallography that determines the electron density distribution of a sample, electron microscopy yields an accurate representation of molecular densities through the projections of the Coulomb potential for a specimen that has been imaged [27]. The density maps are produced by elastic electron interactions with the atomic composition of the macromolecular complex investigated.

Three dimensional electron microscopy reconstruction of single particles (3D
cryo-EM) has become an essential technique in structural biology and has been used to determine structures of large macromolecules, macromolecular complexes and cell components involved in many biological processes including signal transduction, genome replication, transcription and viral infection [2]. The resolution level of the cryo-EM techniques has been improved significantly over time, and today density maps can be produced with a resolution distance below 3 Å [16].

The 3D cryo-EM techniques are based on the assumption that the macromolecules are isolated, randomly oriented, and have identical structures. The procedure to create the density maps from a biological specimen studied can be summarized as follows. A suspension of molecules is placed on a grid, rapidly frozen, and transferred to the microscope. Then, a single exposure picture of a section of a grid is taken, yielding a micrograph filled with hundreds of projections of macromolecules frozen in various orientations. After that, multiple micrographs are collected, individual projections are selected from micrographs, and alignment procedures are used to determine the relative orientations in 3D space. Finally the 3D density distribution is calculated using a 3D reconstruction algorithm.

The Electron Microscopy Data Bank (EMDB) [2] is the major repository for 3D density maps obtained using electron microscopy [28]. The EMDB was created in 2002 by the Macromolecular Structure Database group at the European Bioinformatics Institute (EBI) [4]. Each EMDB entry holds a single density map plus an associated experimental metadata. Recently, a web-based query tool called EMSEARCH (http://emsearch.rutgers.edu/) was created to promote the database exploration using the web.

In order to unify data deposition, processing, and retrieval of maps and fitted models, the EMDB and the Protein Data Bank [3] were unified creating the Unified Data Resource for cryo-EM [2]. Its mission is to build up a global deposition and retrieval network for cryo-EM maps, atomic models and associated metadata, as
Figure 2.2: Density Map for GroEL.
(a) A surface rendering of a single particle reconstruction of GroEL (EMDB access code 1080 [29]) with resolution distance of 11.5 Å and (b) the central slice.

well as a portal for software tools for standardized map format conversion, map segmentation and model assessment, and visualization and data integration. Figure 2.2 shows an example of data that can be obtained from the Unified Data Resource for cryo-EM. Figure 2.2(a) presents a surface rendering for a density map obtained by single particle reconstruction for GroEL (EMBD access code 1080) at resolution distance of 11.5 Å [29] while Figure 2.2(b) shows the central slice.

2.1.2 High-Resolution Structural Information

X-ray crystallography and NMR techniques are two classical tools to produce high-resolution structures. These structures are needed to study the function and evolution of macromolecular assemblies, such as ribosomes, viruses, ion channels, and chaperones [6]. High-resolution structure information has been explored in several research areas such as molecule interaction simulation [30], generating artificially-generated test data (known as phantom data) to test the strengths and limitations of algorithms [13], and drug design [31].

The Protein Data Bank (PDB) [3, 32] is an international repository for 3D structures of macromolecular complexes of proteins, nucleic acids, and other biological molecules. The PDB and its standard input format, the PDB archive, were created in 1971 and first housed by the Brookhaven National Laboratories. In 2003, two
other PDB repositories, the Protein Data Bank Japan (PDBj) [5] and the Macromolecular Structure Database at the European Bioinformatics Institute (PDBe) [4], joined PDB to create the Worldwide PDB (wwPDB) [32, 33]. The wwPDB provides free and public access to the scientific community and ensures that the PDB is a single archive of macromolecular structural data.

A PDB file is a standard storage file for biological atomic structural information, and it is composed of two mains sections: a header and atomic coordinates. The header uniquely identifies a PDB entry and gives details of the experiments that produced this entry. The coordinates section gives a spatial organization of the atoms that compose an atomic structure. The PDB file provides spatial coordinates, an atom name, atomic number, and other chemical and physical properties for each atom in the structure.

However, the PDB file has limitations when representing large structures. For example, the maximum number of atoms in a structure or residues in chain is limited in a PDB file. In this case, the entry can be split into two PDB files or other more versatile representations of structural data could be used, such as nunCIF [35] or PDBML [36]. Figure 2.3 shows the ribbon diagram (i.e., a commonly-used 3D schematic representation of a protein structure) and the PDB file for the bacterial chaperonin GroEL at 2.8 Å (PDB ID 1GRL [34]).

2.1.3 Generation of Density Maps from Atomic Structures

As presented in the previous subsection, the high-resolution structural information is organized as spatial coordinates in a PDB file. The basic idea for converting an atomic model into a density map is to use the coordinates of each atom in the PDB file to compute the intensity for each voxel in the map. This value can be computed based on the atomic number, on the atomic mass, or on the interaction between the atoms in the atomic structure. The computed density maps can be used
Figure 2.3: The Ribbon Diagram and PDB File for the Atomic Model of GroEL at 2.8 Å.

(a) The ribbon diagram for the atomic model of bacterial chaperonin GroEL at 2.8 Å [34] (PDB ID 1GRL); rendering using Jmol (http://www.jmol.org/). (b) Part of the PDB file (access code 1GRL) for bacterial chaperonin GroEL.

There are several software packages (BSoft [18], Situs [37], Xmipp [13, 38, 39], for example) available to convert atomic models into density maps. The purpose of these different software programs is to read a list of atomic coordinates from a PDB file and then compute a density map. In some cases, these programs need some additional information such as the atomic mass or atomic number to compute a density map.
A very simple approach is presented by Heymann [18]: a Gaussian sphere is created at every atomic coordinate and all the spheres are combined to create the final density maps. The density for a voxel in the position \( r \) is computed using the expression

\[
\rho(r) = \sum_i \frac{m_i}{(\sqrt{2\pi\Upsilon})^3} e^{-1/2(|r-r_i|/\Upsilon)^2},
\]

where \( r_i \) is the location of the \( i \)th atom, \(|r - r_i|\) is the distance between the position \( r \) and \( r_i \), \( \Upsilon \) is half the desired resolution distance, and \( m_i \) is the atomic weight (or mass in Daltons) for the atom \( i \). This methodology is currently available in BSoft [18] – a collection of programs and a platform for development of software for image and molecular processing in structural biology.

A more elaborate approach is presented in Wriggers [37], where a simulated density map is created from atomic coordinates by real-space kernel convolution. In this approach, the atomic coordinates are first projected onto a regular cubic grid by trilinear interpolation, and then each grid point is convolved with one of the supported kernel functions. As examples of the smoothing kernels available, we have:

\[
\text{gaussian}(r) = e^{-\frac{1.5r^2}{\zeta^2}},
\]

\[
\text{triangular}(r) = \max\left(0, 1 - \frac{0.5|r|}{\zeta}\right),
\]

\[
\text{epanechnikov}(r) = \max\left(0, 1 - \frac{0.5r^2}{\zeta^2}\right),
\]

\[
\text{hardphere}(r) = \max\left(0, 1 - \frac{0.5r^{60}}{\zeta^{60}}\right),
\]

where \( \zeta \) is the half-max kernel radius. This methodology is implemented in Situs.
Sorzano et al. [13] introduced a fast and accurate real space method to convert atomic models into density maps based on Electron Atomic Scattering Factors (EASF). An atomic scattering factor, or atomic form factor, is a measure of the scattering amplitude of a wave (in this case, the wave produced by the electron) in an isolated atom. The EASF of each atom has been studied and measured experimentally and can be accurately approximated by a sum of Gaussians up to a frequency of 6 Å⁻¹ using the following equation

\[ F_e(R) = \sum_{i=1}^{n} a_i \exp(-b_i R^2), \]  

(2.6)

where \( F_e(R) \) is the electron scattering factor of a given atom at the spatial frequency \( R \) (in Å⁻¹) [40]. The parameters \( a_i \) and \( b_i \) are specific for each atom. The EASF in real space can be expressed by

\[ f_e(r) = \sum_{i=1}^{n} 2\pi a_i \sqrt{\frac{\pi}{b_i}} \exp\left(-\frac{r^2 z_i^2}{b_i}\right). \]  

(2.7)

To compute a density map for a PDB atomic model, the position of each atom in the list of atomic coordinates is substituted by the corresponding EASF using the following equation

\[ \rho_{easf}(r) = \sum_{i=1}^{n} f_e(|r - r_i|), \]  

(2.8)

where model \( \rho_{easf} \) is the continuous function representing the atomic model of the macromolecule, \( r \in \mathbb{R}^3 \) is the coordinate at which the volume (here, and in many places in the following, the word “volume” is used in the sense of a “3D image”) is being evaluated, \( r_i \) is the center of the \( i \)th atom and \( f_{e_i} \) is the EASF of the \( i \)th atom listed in the PDB file. The function \( \rho_{easf} \) is a continuous function in \( \mathbb{R}^3 \) and therefore needs to be sampled in order to be computational represented and
processed. After this function is sampled, it can be down-sampled by applying a low-pass filter and a decimation operator.

Sorzano et al. [13] propose to combine the sampling, low-pass filtration, and decimation into one single step. The proposed low-pass filtered version of the EASF, the Low-pass Electron Atomic Scattering Factors (LEASF), is the effective band-limited function and can be safely sampled at the final sampling rate without any significant aliasing. The atomic model can be computed using the new LEASF by

$$\tilde{\rho}_{easf}(r) = \sum_{i=1}^{n} \tilde{f}_{ei}(|r-r_i|)$$

(2.9)

where $\tilde{f}_{ei}$ is the LEASF of the $i$th atom listed in the PDB file.

### 2.2 Docking High-Resolution Structures into a Low-Resolution Density Map

Combining low-resolution density maps obtained from cryo-EM with high-resolution atomic structures obtained from X-ray crystallography or NMR spectrography has been used to understand the function and evolution of macromolecular complexes. Finding the position of these atomic structures, which usually represent subunits in a whole macromolecular complex, has proved to be useful for understanding conformation changes at near atomic-level details in several biological systems [7]. The process of combining low-resolution images with high-resolution images is known as the docking or fitting of the atomic structure into a density map.

Two early demonstrations of docking were done in the study of viruses [41] and muscles [42]. For [41], the authors investigated particles of adenovirus type 2 and localized a minor component of GON using a combination of electron microscopy and X-ray crystallography. For two brief historical overviews of macromolecular
STEP 1 - a PDB file with the atomic coordinates for an atomic structure and a low-resolution density map are received as input. STEP 2 - a high-resolution density map is computed from the PDB file. STEP 3 - the position and orientation of the high-resolution density map in the low-resolution density map is found by optimizing some quality-of-fit measure.

There are some issues that need to be observed in the process of docking a subunit into the low-resolution density map. The first issue is that a low-resolution density map may not have sufficient features for an unambiguous subunit placement. In these cases, multiple different positions of the subunit yield similar fitness scores. If available, additional biochemical or biophysical information could help
in deciding the final position of the subunit. The second issue is that subunits can have different structural conformations due to interaction with the other subunits in the macromolecular assembly. These conformations include shear and hinge movements (two common protein motions), loop distortions, and movements in the secondary structures for example. In addition, the target density map (representing a macromolecular complex) also may have multiple conformation states due to its macromolecular function. To overcome these problems, the docking methodology needs to take care of realistic conformation changes in the subunit and/or the low-resolution density map.

The third issue is the large computation cost in finding the position of one or more subunits into a target density map. A methodology that tries to fit $n$ subunits into a low-resolution density map one-by-one, deals with six degrees of freedom in the search space. Besides the high computation cost for this search, a sequential strategy may not find an optimal fit because the position of the first subunit is not modified by the following optimization. In the following paragraphs, we describe four approaches to docking a high-resolution density map into a low-resolution density map.

Initially, the docking of an atomic model into a low-resolution density map was performed manually. In this approach, called manual docking, a user – typically an expert in the field – interacts with a visualization tool to place the atomic structure into the low-resolution density map. This is an exhaustive and time consuming process and is strongly based on the knowledge of the user. Despite the inherent subjective task of manual docking, significant results were obtained, as reported in [44]. For manual docking, success is measured by the satisfaction of the user and could be easily contested by other professionals. See [41, 42] for more details about manual docking.
2.2.1 Rigid-body Docking

A quantitative approach, known as rigid-body docking, is based on a voxel-wise comparison of densities between the high-resolution density map and the low-resolution density map. This quantitative approach relies on a systematic search over three translational and three rotational degrees of freedom in real space. In the rigid-body docking approach, the target and the model are considered as rigid bodies and no conformational states are taken into consideration. In the rigid-body docking, the task is to find the position and orientation of the model into the target optimizing some quality-of-fit measure. A short review of these methods can be found in [7, 37].

The most popular quality-of-fit measure used in rigid-body techniques is the cross-correlation coefficient. The cross-correlation is a well-known measure of similarity. Let $I_t : V_t \to \mathbb{R}$ be a target density map and $I_m : V_m \to \mathbb{R}$ represent a model density map. Then, the cross-correlation is computed by the equation

$$CC = \frac{\sum_{i=1}^{M} (I_t(i) - \bar{I}_t) (I_m(i) - \bar{I}_m)}{\sigma_I \sigma_{I_m}},$$

(2.10)

where $M$ is the number of grid points in the model density map, $\sigma_I$ and $\bar{I}_t$ are the standard deviation and mean of $I_t$ respectively, and $\sigma_{I_m}$ and $\bar{I}_m$ are the standard deviation and mean of $I_m$ respectively. Here we assume that the two density maps are on the same grid and that all voxels of $I_t$ are also in $I_m$. The larger the value of the cross-correlation coefficient, the more correlated the signals are.

Now consider two density maps with two different grids: a high-resolution-grid density map $I_m$ covering a small region and a low-resolution-grid density map $I_t$ covering a large region. Every translation and rotation of $I_m$ into the region covered by $I_t$ will give a configuration of high-resolution grid points inside the low-resolution-grid region. Values at each of these high-resolution grid points can be
estimated from the low-resolution density map $I_m$ by interpolation. After the interpolation process, the cross-correlation can be calculated by comparing the high-resolution density map $I_t$ with the image for the same grid whose values we obtained by the interpolation.

Let $x$ be a position among all the possible positions of the high-resolution density map $I_m$ in the low-resolution density map $I_t$. The following equation can be used to compute the best position of $I_m$ in $I_t$

$$CC(x) = \frac{\sum_{i=1}^{M} (I_t(i) - \bar{I}_t)(I_m(i + x) - \bar{I}_m)}{\sigma_{I_t} \sigma_{I_m}},$$

(2.11)

where $M$ is the number of grid points in the high-resolution density map, $\sigma_{I_t}$ and $\bar{I}_t$ are the standard deviation and mean of $I_t$ respectively, and $\sigma_{I_m}$, $\bar{I}_m$ are the standard deviation and mean of $I_m$ after the interpolation process explained above. It is important to mention that the difference between Equations 2.10 and 2.11 is that for the former, $I_m$ and $I_t$ have the same grid and all voxels of $I_t$ are also in $I_m$, while the latter computes the correlation for a specific position of $I_m$ in $I_t$ defined by $x$.

Clearly the issue of six degrees of freedom in the search space is a challenge to be overcome for such docking methodologies. The computational cost will be much higher if the task is not docking only one subunit, but instead independently fitting several subunits in the same target density map. Besides the higher computational cost, there are other issues that increase the complexity of the docking. One such example is that there are insufficient features in the target density map for an unambiguous placement of the model and the different conformations in the atomic structure and in the target density map. In addition, the target and/or model may be deformed by the experimental procedure or by errors in the imaging process.

Equations 2.10 and 2.11 describe how a global cross-correlation can be used to find the position of the model density maps in a target density map. As is dis-
cussed in [11], normalization in the cross-correlation is of critical importance for the success of the search process because the density map intensities can vary due to the process of producing them. However, the use of global normalization to find the position of the model in the target map can fail since a correlation peak can be found in a local density maximal in the target density map.

To address this problem, a local normalization can be used to compute the cross-correlation. The following locally variable normalization to compute the cross-correlation is presented in [7]

\[
CC_L(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{(I_m(i) - \bar{I}_m)M(i)(I_t(i+x) - \bar{I}_t)}{\sigma_{I_m} \sigma_{M I_t}(x)},
\]

(2.12)

where \(I_m\) is the model density map and \(I_t\) is the target density map, \(N\) is the number of voxels on \(I_m\), and \(M(i)\) is a mask function that defines the boundary of the search object defined by \(I_m\). The product \(M(i)I_t(i+x)\) defines the 3D region on the target density map as the model density map assumes all possible positions symbolically indicated by \(x\). \(\bar{I}_m\) and \(\bar{I}_t\) are the mean of \(I_m\) and \(I_t\), and \(\sigma_{I_m}\) and \(\sigma_{M I_t}\) are the local variance of two densities within the exact region of intersection in the current relative position. In \(CC_L\), it is assumed that the function mask \(M\) takes care of the interpolation and the resolution issues mentioned in Equation 2.11.

One of the main drawbacks for computing rigid docking using cross-correlation is the computational cost to find the position of the atomic model in the six-dimensional search space. There are several strategies that attempt to reduce the computation time for rigid-body docking. As example, we can cite [44] that computes the correlation in the reciprocal space; [6] where the correlation is computed over the small block of the density map; [45] that uses a Laplacian filter combined with a six-dimensional search using fast Fourier transform; and [46] the use of vector quantization to reduce the search space.
2.2.2 Flexible Docking

In the rigid-body docking approach, it is assumed that the objects imaged, whole complex macromolecules and their subunits, are rigid bodies. Unfortunately, this is not true for all the biological species studied. The flexible docking methodologies take into consideration the conformation changes that happen in the target and in the model density maps. Flexible docking is essential for a molecular-level understanding of conformational changes and their possible biological relevance. Tama et al. [47] stated that in some cases, the conformational reorganization of the high-resolution structure may be necessary to rationalize the conformation observed in the cryo-EM experiments.

A typical example is when the structure of a subunit is modified after the interaction with the other subunits in the whole macromolecular complex. If the atomic model was obtained by imaging the isolated subunit, the flexible docking approach attempts to reproduce the structural changes in the atomic model in order to closely represent the conformational changes and consequently maximize the quality-of-fit measure.

A common practice in the flexible docking is to subdivide the process in two phases: reduce the search space and refine the initial placement. In the first phase, the model density map is rigidly fitted into an approximate position and orientation in the density map. This can be done manually or using any quantitative rigid-body docking procedure. Once positioned, the flexible docking methodology refines this initial position by the maximization of a quality-of-fit measure.

Different flexible docking strategies have been proposed. The idea is to predict what deformation should be applied to the model density map to best represent the structural conformation of the subunit into the target density map. As examples, we have [30] that uses molecular dynamics simulation to compute the conformation changes; [47, 48] that apply linear combination of low-frequency Normal Modes to
deform the atomic structure; and [49] that applies Monte Carlo simulations to maximize the cross-correlation coefficient and to simulate the motion of the biomolecule as rigid cluster. Some authors use an additional level of information to predict the structural deformation. For example, [49] use stereochemistry of the model and the nonbonded atom-atom contacts, [44] uses a set of common molecular symmetries, and [30] that uses the atomic mass and charge.

### 2.2.3 Multiple Docking

The last docking strategy addressed here is called *multiple docking*. Consider the scenario where we need to fit a set of subunits to a macromolecular complex. One could plan to dock all the subunits one-by-one within a sequential strategy, starting from the largest one until the smallest subunit. Besides the computational cost involved in the sequential strategy, the final result cannot be satisfactory if the remaining subunits cannot be unambiguously fitted into the unoccupied macromolecule. Zhang *et al.* [50] say that sequential fitting of subunits into a density map often fails when the resolution distance of a target the density map is approximately 10 to 30 Å. In addition, Lasker *et al.* [51] explain that this strategy is not adequate when the component models are inaccurate and the number of subunits is large.

Several methodologies have been proposed to simultaneously dock subunits into target density map. In [51], a combinatorial optimization protocol is used with a quality-of-fit measure composed by a correlation coefficient and two geometric information. The combinatorial problem is reformulated as an inferential optimization over a discrete sample space using a methodology called MultiFit. In [50], a vector quantization is used to limit the search space and a quadratic programming strategy is used speed up the computation.

In the work presented in [52], the resolution of the subunit density map is reduced to the same resolution of the target density map. The author uses the Gaussian
Mixture Model to represent the subunits and target density maps. He claims that the Gaussian distribution function can approximate the geometry of the complicated atomic structures and target maps and improve the computational time. However, the performance is directly related to the number of functions used to represent the model and target map. The higher the number of functions, the higher the computation cost.

### 2.3 The Embedded Tree Problem and Its Algorithms

The biomolecular docking task is to find where the high-resolution model density map is within the low-resolution target density map. It is conceivable that conversion of density maps into a different data structure would reduce the computational burden of molecular docking. As a part of the macromolecular exploration process, we propose to use a tree data structure, known as a component tree, to accomplish the docking task. In particular, we investigate the relationship between component trees obtained from the target and the model density maps. As will be discussed in Section 5.1, this relationship is likely to be one of embedding. In what follows, we present a definition of tree embedding and a brief review of methodologies to find embedded trees.

Graphs, especially trees, have been used widely to represent data and their relationships in various problem domains such as web mining, XML documents mining, bioinformatics, and social networks. Several applications and algorithms using trees as data structures can be found in [53, 54, 55, 56, 57]. This dissertation will investigate the use of component trees to represent relationships in the set of connected components produced from 3D density maps. In particular, we use the embedded subtree relationship between component trees to analyze the structural relationship between their respective density maps.
A rooted tree $T$ is a directed acyclic connected graph represented by a pair $(N,E)$, where $N$ is a nonempty finite set of nodes and $E$ is a set of edges. Each edge is an ordered pair of distinct nodes that are respectively called the parent node and the child node of the edge, and the nodes and edges satisfy the following conditions: (1) every member of $N$, except one element called the root, is a child node of just one edge; (2) the reflexive transitive closure of $E$ is a partial order on $N$. A tree is called a labeled rooted tree if $\Omega : N \rightarrow L$ is a labeling function mapping nodes into a set of labels $L = \{l_1, l_2, \ldots \}$. An ordered labeled rooted tree is a labeled rooted tree where the children of each internal node are ordered based in their labels. If the children are not ordered we call this labeled rooted tree of unordered labeled rooted tree.

Let $T = (N,E)$ be a rooted tree. If $C \in N$ and $D$ is a node of the subtree of $T$ that is rooted at $C$, then $C$ is said to be an ancestor of $D$ in $T$ and $D$ a descendant of $D$ in $T$. We write $C \leq_T D$ or $D \geq_T C$ to mean that $C,D \in N$ and $C$ is an ancestor of $D$ in $T$. We write $C \prec_T D$ or $D \succ_T C$ to mean that $C \leq_T D$ but $C \neq D$. If $C \prec_T D$, then $C$ is said to be a proper ancestor of $D$ in $T$ and $D$ a proper descendant of $C$ in $T$.

Given two rooted trees $T_1 = (N_1,E_1)$, and $T_2 = (N_2,E_2)$, we say that $T_1$ is an isomorphic subtree of $T_2$ if, and only if, there exist a one-to-one mapping $\varphi_i : N_1 \rightarrow N_2$, such that $(C,D) \in E_1$ if, and only if, $(\varphi_i(C), \varphi_i(D)) \in E_2$. In the case when $\varphi_i$ is onto, then we say that $T_1$ and $T_2$ are isomorphic.

A tree embedding of a rooted tree $T_1 = (N_1,E_1)$ into a rooted tree $T_2 = (N_2,E_2)$ is a map $\varphi_e : N_1 \rightarrow N_2$ such that $C$ is a descendant of $D$ in $T_1$ if, and only if, $\varphi_e(C)$ is a descendant of $\varphi_e(D)$ in $T_2$. The map $\varphi_e$ that defines a tree embedding of $T_1$ into $T_2$ is called an embedding map. Note that every tree embedding $\varphi_e$ is a 1-to-1 mapping, because if $\varphi_e(C') = \varphi_e(D')$ then each of $\varphi_e(C)$ and $\varphi_e(D)$ is a descendant of the other, whence each of $C$ and $D$ is a descendant of the other, which implies
$C = D$. It is important to note that the ancestor relationship is preserved under tree embeddings.

Using the definition above, the tree embedding problem can be stated as follows: given two trees $T_1$ and $T_2$, list all the embedding maps $\varphi_e$ of $T_1$ into $T_2$ or decide that there is no such map. There are several algorithms in the literature for the tree embedding problem. We are interested in investigating algorithms for finding embedding maps of unordered labeled rooted trees. Before giving some examples of algorithms proposed to solve the tree embedding problem, we discuss a very relevant fact about the tree embedding problem in the next paragraph.

In a paper by Kilpelainen and Mannile [58], it is proven that the tree embedding problem for unordered labeled rooted trees is a NP-complete problem. This proof is based on showing that the tree embedding problem is equivalent to the NP-complete problem called the inclusion tree problem.

Fortunately, in many practical situations, a polynomial time algorithm can be found for the unordered inclusion problem. A practical example is when we are interested in finding an included tree $T_1$ in $T_2$, in which $T_2$ is a large database. The first polynomial time solution was presented by [58]. The authors showed that if the out-degree on the node of $T_2$ is bounded by $k$, the unordered tree inclusion problem can be solved in $O(|T_1|k2^{2k}|T_2|)$. More specifically, if $k$ is $O(\log |T_2|)$, the problem is solvable in time $O(|T_1|\log |T_2||T_2|^3)$.

Frequent Tree Pattern Mining is another example where the embedded subtrees are computed in polynomial time. One of the tasks of the mining problem is to identify frequent embedded subtrees in a database of ordered labeled trees [53]. Let $F = \{T_1, T_2, \ldots, T_n\}$ be a collection of trees that compose the database $D$. The task of identify frequent embedded subtrees can be done in two steps. First, list all subtrees $T' = (N', E')$ of $T_i = (N_i, E_i)$ such that there exists an embedding map $\varphi_e : N' \rightarrow N_i$, for $1 \leq i \leq n$. In the second step, the frequency of all embedded
subtrees in the database $D$ is computed. For examples of algorithms that compute frequent embedded subtrees, see [56, 57, 59].

In Chapter 5, we investigate the use of embedding maps to accomplish the macromolecular docking task. We expect be able to design an algorithmic solution to compute embedding maps for our component trees.
Chapter 3

Digital Pictures and their Component Trees

In this chapter we summarize, in separate sections, two published works [19, 20]. These papers give theoretical support to the methods presented in this dissertation.

The first work presents a mathematical theory of the intuitive concept of embedded pictures and connectedly embedded pictures [19]. When a picture is connectedly embedded in another picture, a stable relationship between the component trees (formally defined later in this chapter) of these two pictures exists. This theory can be useful in understanding the relation between two pictures of the same object at different resolutions.

In the second work, a careful investigation is done to explore the potential use of component trees to represent biological specimens [20]. After defining a foreground component tree structure, the paper explores the relationship between pictures and their component trees. A robust simplification methodology for component trees is also presented. Lastly, an experiment using foreground component tree structures to analyze two different macromolecules is provided.
3.1 Tree Representation of Digital Picture Embeddings

In what follows, a mathematical theory of the intuitive concept of embedded pictures is presented. In many applications, an object is imaged in different ways resulting in digital pictures of (some parts of) it at different resolutions. A prime example arises in structural biology when cryo-EM and X-ray have been used as complementary methods to explore biological information. This section presents a basis for a theory of digital picture embeddings, motivated by applications of image processing when the same object is imaged at different resolutions.

3.1.1 Embedded Digital Spaces and Pictures

In what follows, we use V to denote a nonempty finite set and π to denote a symmetric irreflexive binary relation on V. If (c, d) ∈ π, then we say that c and d are π-adjacent. Elements of V will be called spells (as in, e.g., [60], where “spel” is an abbreviation for “spatial element”) and we think of spells as generalizations of pixels and voxels.

Let A be a subset of V. For any c and d in A, the sequence \( \langle d^{(0)}, \ldots, d^{(K)} \rangle \) of elements of A is said to be a π-path in A connecting c to d, if \( d^{(0)} = c, \  d^{(K)} = d, \) and, for \( 0 \leq k < K, \ d^{(k)} \) is π-adjacent to \( d^{(k+1)} \). If there is a π-path in A connecting c to d, then we say that c is π-connected in A to d. (By setting \( K = 0 \) and considering the sequence \( \langle c \rangle \), we see that c is π-connected in A to c, for any c in A.) A subset A of V is said to be a π-connected if, for any c and d in A, c is π-connected in A to d. (According to this definition, the empty set is π-connected and the set \{c\} is π-connected, for any c in V.) If V is π-connected, then we call the pair \((V, \pi)\) a digital space.

A digital picture over the digital space \((V, \pi)\) is a triple \((V, \pi, f)\), where f is
a function that maps $V$ into the real numbers [60]. For each $c \in V$ we say that $f(c)$ is the intensity level (or density level or graylevel) of $c$. We say that a digital picture $(V_1, \pi_1, f_1)$ is isomorphic to a digital picture $(V_2, \pi_2, f_2)$ if there is a bijective correspondence $\xi : V_1 \to V_2$ such that, for all $x_1, y_1 \in V_1$: (i) $f_2(\xi(x_1)) = f_1(x_1)$ and (ii) $(\xi(x_1), \xi(y_1)) \in \pi_2$ just if $(x_1, y_1) \in \pi_1$. For any real number $t$, the $t$-superlevel set of $(V, \pi, f)$ is $V_{f,t} = \{ c \in V \mid t \leq f(c) \}$ [61].

Let $A$ and $C$ be subsets of $V$. We say that $C$ is a $\pi$-component of $A$ if

1. $C$ is a nonempty $\pi$-connected subset of $A$ and

2. for all $\pi$-connected subsets $B$ of $A$, if $C \subseteq B$, then $B = C$.

**Proposition 1.** For any digital picture $(V, \pi, f)$ and any $c \in V$, there is a unique $\pi$-component of $V_{f,f(c)}$ that contains $c$ and it is given by

$$C_{(V,\pi,f)}(c) = \{ d \in V \mid c \text{ is } \pi\text{-connected in } V_{f,f(c)} \text{ to } d \}. \quad (3.1)$$

**Proof.** Clearly, $C_{(V,\pi,f)}(c)$ as defined in Eq. (3.1) contains $c$. Next we show that it is a $\pi$-component of $V_{f,f(c)}$:

$C_{(V,\pi,f)}(c)$ is nonempty. If $d$ is in $C_{(V,\pi,f)}(c)$, then there is a $\pi$-path $\langle d^{(0)}, \ldots, d^{(K)} \rangle$ such that $d^{(0)} = c$, $d^{(K)} = d$, and, for $0 \leq k \leq K$, $f(c) \leq f\left(d^{(k)}\right)$. In particular, $f(c) \leq f(d)$, implying that $C_{(V,\pi,f)}(c)$ is a subset of $V_{f,f(c)}$. We note that it follows from (3.1) that, for $0 \leq k \leq K$, $d^{(k)} \in C_{(V,\pi,f)}(c)$. Now let $e$ be also an element of $C_{(V,\pi,f)}(c)$. By the same argument, there is a $\pi$-path $\langle e^{(0)}, \ldots, e^{(L)} \rangle$ such that $e^{(0)} = c$, $e^{(L)} = e$, and, for $0 \leq \ell \leq L$, $e^{(\ell)} \in C_{(V,\pi,f)}(c)$. By considering the $\pi$-path $\langle d = d^{(K)}, \ldots, d^{(0)} = c = e^{(0)}, \ldots, e^{(L)} = e \rangle$, we see that $d$ is $\pi$-connected in $C_{(V,\pi,f)}(c)$ to $e$ and so $C_{(V,\pi,f)}(c)$ is $\pi$-connected. So far we have proved that $C_{(V,\pi,f)}(c)$ is a nonempty $\pi$-connected subset of $V_{f,f(c)}$. Now let $B$ be a $\pi$-connected subset of $V_{f,f(c)}$ such that $C_{(V,\pi,f)}(c) \subseteq B$. It follows that $c$ is in $B$ and, for any $b$ in $B$, $c$ is $\pi$-connected in $B \subseteq V_{f,f(c)}$ to $b$. By (3.1) this implies that $b$ is in $C_{(V,\pi,f)}(c)$.
and so \( B \subseteq C_{(V, \pi, f)}(c) \) and, in fact, \( B = C_{(V, \pi, f)}(c) \). All this proves that \( C_{(V, \pi, f)}(c) \) is a \( \pi \)-component of \( V_{f, f(c)} \).

Now suppose that \( D \) is a \( \pi \)-component of \( V_{f, f(c)} \) that contains \( c \). Let \( B = C_{(V, \pi, f)}(c) \cup D \). Clearly, \( B \) is a subset of \( V_{f, f(c)} \). Since \( c \) is in both \( C_{(V, \pi, f)}(c) \) and \( D \), for any \( d \) and \( e \) in \( B \), we can combine (as in the previous paragraph) the \( \pi \)-paths in \( B \) connecting \( d \) to \( c \) and \( c \) to \( b \), to provide a \( \pi \)-path in \( B \) connecting \( d \) to \( e \). Hence, \( B \) is a \( \pi \)-connected subset of \( V_{f, f(c)} \). From the fact that the \( \pi \)-component \( C_{(V, \pi, f)}(c) \) is a subset of \( B \), it follows that \( B = C_{(V, \pi, f)}(c) \). From the fact that the \( \pi \)-component \( D \) is a subset of \( B \), it follows that \( B = D \). Hence \( D = C_{(V, \pi, f)}(c) \) and \( C_{(V, \pi, f)}(c) \) is the unique \( \pi \)-component of \( V_{f, f(c)} \) that contains \( c \).

The concepts introduced so far are illustrated in Figure 3.1. The same figure illustrates the following proposition as well.

**Proposition 2.** For any digital picture \((V, \pi, f)\) and any \( c \) and \( d \) in \( V \), such that \( f(c) \leq f(d) \) and \( d \in C_{(V, \pi, f)}(c) \), it is the case that \( C_{(V, \pi, f)}(d) \) is a subset of \( C_{(V, \pi, f)}(c) \).

**Proof.** The assumption that \( d \in C_{(V, \pi, f)}(c) \), implies that there exists a \( \pi \)-path \( \langle d^{(0)}, \ldots, d^{(K)} \rangle \) such that \( d^{(0)} = c \), \( d^{(K)} = d \) and, for \( 0 \leq k \leq K \), \( f(c) \leq f(d^{(k)}) \). We are now going to prove that if \( e \in C_{(V, \pi, f)}(d) \), then \( e \in C_{(V, \pi, f)}(c) \). If \( e \in C_{(V, \pi, f)}(d) \), then here exists a \( \pi \)-path \( \langle e^{(0)}, \ldots, e^{(L)} \rangle \) such that \( e^{(0)} = d \), \( e^{(L)} = e \) and, for \( 0 \leq \ell \leq L \), \( f(d) \leq f(e^{(\ell)}) \). Recalling the assumption that \( f(c) \leq f(d) \) and considering the \( \pi \)-path \( \langle c = d^{(0)}, \ldots, d^{(K)} = d = e^{(0)}, \ldots, e^{(L)} = e \rangle \), we see that \( c \) is \( \pi \)-connected in \( V_{f, f(c)} \) to \( e \) and so (3.1) implies that \( e \in C_{(V, \pi, f)}(c) \).

The set of components of \((V, \pi, f)\) is

\[
\mathcal{C}_{(V, \pi, f)} = \{ C_{(V, \pi, f)}(c) \mid c \in V \}. \tag{3.2}
\]

It is readily confirmed that \( V \in \mathcal{C}_{(V, \pi, f)} \) and that any two members of \( \mathcal{C}_{(V, \pi, f)} \) are
That prove the converse, we assume that \( \min f \). For the pixel \( p \), \( f(p) = 1 \) and \( V_{f,p} = V_{f,1} \) is the set of all light and dark gray pixels. Since this set is \( \pi \)-connected, \( C_{V,\pi,f}(p) = V_{f,p} \). For the pixel \( q \), \( f(q) = 2 \) and \( V_{f,q} = V_{f,2} \) is the set of all dark gray pixels. Since this set is not \( \pi \)-connected, \( C_{V,\pi,f}(q) \neq V_{f,q} \); in fact, \( C_{V,\pi,f}(q) \) comprises the nine dark gray pixels on the left side of the image. Note that \( C_{V,\pi,f}(q) \) is a subset of \( C_{V,\pi,f}(p) \) since \( f(p) \leq f(q) \) and \( q \in C_{V,\pi,f}(p) \).

disjoint unless one is a subset of the other. The members of \( \mathcal{C}_{V,\pi,f} \) are sometimes called \textit{maximum intensity extremal regions} [62].

For the digital picture of Figure 3.1, the members of \( \mathcal{C}_{V,\pi,f} \) are \( V, P = C_{V,\pi,f}(p), Q = C_{V,\pi,f}(q), R = C_{V,\pi,f}(r) \) and \( U = C_{V,\pi,f}(u) \); \( P \) consists of all the light gray and dark gray pixels, \( Q \) consists of the nine dark gray pixels on the left, \( R = \{r\} \), and \( U \) consists of the twenty dark gray pixels on the right. For the pixel \( s, s \in P, f(s) = \min \{ f(e) \mid e \in P \} \) and \( P = C_{V,\pi,f}(s) \), which illustrates the following proposition.

\textbf{Proposition 3.} For any digital picture \((V,\pi,f)\), any \( C \in \mathcal{C}_{V,\pi,f} \) and any \( c \in C, f(c) = \min \{ f(e) \mid e \in C \} \) if, and only if, \( C = C_{V,\pi,f}(c) \).

\textit{Proof.} If \( C = C_{V,\pi,f}(c) \), then \( c \in C \) and so \( \min \{ f(e) \mid e \in C \} \leq f(c) \). On the other hand, for every \( e \in C, e \) is in the \( f(c) \)-superlevel set of \((V,\pi,f)\) and so \( f(c) \leq \min \{ f(e) \mid e \in C \} \). This proves that \( f(c) = \min \{ f(e) \mid e \in C \} \) if \( C = C_{V,\pi,f}(c) \). To prove the converse, we assume that \( f(c) = \min \{ f(e) \mid e \in C \} \) and prove the result that \( C = C_{V,\pi,f}(c) \).
Since $C \in \mathcal{C}_{(V, \pi, f)}$, there must be a $d \in V$ such that $C = C_{(V, \pi, f)}(d)$. We see from the “if” part of this proposition (which was established in the previous paragraph) that $f(d) = \min \{ f(e) \mid e \in C \} = f(c)$, and so it follows from Proposition 1 that $C$ is a $\pi$-component of $V_{f, f(d)} = V_{f, f(c)}$. Since $C$ is a $\pi$-component of $V_{f, f(c)}$ and $c \in C$, it follows from Proposition 1 that $C = C_{(V, \pi, f)}(c)$. □

A digital space $(V', \pi')$ is said to be embedded in the digital space $(V, \pi)$ if

1. $V'$ is a partition of $V$,

2. every $c' \in V'$ is a $\pi$-connected subset of $V$, and

3. for all $c'$ and $d'$ in $V'$, $(c', d') \in \pi'$ if, and only if, $c' \neq d'$ and there exist a $c \in c'$ and a $d \in d'$ such that $(c, d) \in \pi$.

A digital picture $(V', \pi', f')$ is said to be embedded in the digital picture $(V, \pi, f)$ if

1. $(V', \pi')$ is embedded in $(V, \pi)$ and

2. for all $c' \in V'$,

$$f'(c') = \min \{ f(c) \mid c \in c' \}.$$ \hspace{1cm} (3.3)

In these two definitions, if $V'$ is a partition of $V$ whose elements all have the same cardinality (i.e., each element of $V'$ consists of $\frac{|V|}{|V'|}$ elements of $V$), then we say that the digital space $(V', \pi')$ or the digital picture $(V', \pi', f')$ is regularly embedded in the digital space $(V, \pi)$ or the digital picture $(V, \pi, f)$. Figure 3.2 illustrates the concepts of regularly embedded digital space and regularly embedded digital picture.

When considering embedded digital pictures it is often helpful to bear in mind the following fact, which is an immediate consequence of the definition of $C_{(V, \pi, f)}(c)$:
Figure 3.2: Embedded Digital Picture.
(a) shows a digital picture \((V', \pi', f')\) whose digital space \((V', \pi')\) is regularly embedded in the digital space \((V, \pi)\) of Figure 3.1. (b) shows the partition \(V'\) of \(V\) overlaid on the digital picture \((V, \pi, f)\) of Figure 3.1. It is clear from (c) that each element of \(V'\) is a 4 \times 4 subarray of the pixels in \(V\), and it can also be seen that, e.g., \(p \in p' \in V'\), \(q \in p' \subset V'\) and \(r \in s' \in V'\). The digital picture \((V', \pi', f')\) is regularly embedded in the digital picture \((V, \pi, f)\). Since \(f(p) \leq f(c)\) for any \(c \in p'\), \(f'(p') = f(p) = 1\). \(C(V', \pi', f')(p')\) comprises the four light gray elements of \(V'\) on the left side of (a).

Remark 1. Let \((V, \pi, f)\) be a digital picture, \(c'\) be a \(\pi\)-connected subset of \(V\) and \(c \in c'\) be such that \(f(c) = \min \{f(e) \mid e \in c'\}\). Then \(c'\) is a subset of \(C(V, \pi, f)(c)\).

A special case of Remark 1 is that if \((V', \pi', f')\) is embedded in \((V, \pi, f)\) and \(c' \subset V'\), then \(c'\) is a subset of \(C(V, \pi, f)(c)\) for any \(c \subset c'\) such that \(f(c) = \min \{f(e) \mid e \subset c'\}\).

Consider, for example, the digital pictures \((V, \pi, f)\) and \((V', \pi', f')\) from Figures 3.1 and 3.2. Here we have that \(p \in p' \subset V'\) and \(f(p) = 1 = \min \{f(e) \mid e \subset p'\}\), and we see that \(p'\) is indeed a subset of \(C(V, \pi, f)(p)\).

Lemma 1. Let \((V', \pi', f')\) be a digital picture embedded in the digital picture \((V, \pi, f)\), let \(C'\) and \(D'\) be elements of \(C(V, \pi, f')\) such that \(D'\) is a subset of \(C'\) and let \(c' \subset C', d' \subset D', c \subset c'\) and \(d \subset d'\) be such that

\[
f(c) = f'(c') = \min \{f'(e') \mid e' \subset C'\} \quad (3.4)
\]

and

\[
f(d) = f'(d') = \min \{f'(e') \mid e' \subset D'\} \quad (3.5)
\]
Then $C_{(V, \pi, f)}(d)$ is a subset of $C_{(V, \pi, f)}(c)$.

Proof. It follows from the assumption that $D'$ is a subset of $C'$ and from (3.4) and (3.5) that $f(c) = f'(c') \leq f'(d') = f(d)$. It follows from Proposition 2 that all we need to show is that $d \in C_{(V, \pi, f)}(c)$. By Proposition 3, (3.4) and (3.5), $C' = C_{(V', \pi', f')} (c')$ and $D' = C_{(V', \pi', f')} (d')$. Since $D'$ is a subset of $C'$, $c'$ is $\pi'$-connected in $V', f'(c')$ to $d'$. Hence there is a $\pi'$-path $\langle d'^{(0)}, \ldots, d'^{(K)} \rangle$ such that $d'^{(0)} = c'$, $d'^{(K)} = d'$ and, for $0 \leq k \leq K$, $f'(c') \leq f'(d'^{(k)})$. By the definition of $\pi'$, for $0 \leq k < K$, there exist $d'^{(k)} \in d'^{(k)}$ and $c^{(k+1)} \in d'^{(k+1)}$ such that $\langle d'^{(k)}, c^{(k+1)} \rangle \in \pi$. We also define $e^{(0)} = c$ and $d^{(K)} = d$. With these definitions it is the case that, for $0 \leq k \leq K$, both $c^{(k)}$ and $d^{(k)}$ are in $d'^{(k)} \in V'$. By the definition of an embedded digital space, $d'^{(k)}$ is a $\pi$-connected subset of $V$ and so there exists a $\pi$-path $\langle e^{(k,0)}, \ldots, e^{(k,L_k)} \rangle$ in $d'^{(k)}$ such that $e^{(k,0)} = c^{(k)}$ and $e^{(k,L_k)} = d^{(k)}$. Furthermore, for $0 \leq \ell \leq L_k$, $f\left(\langle e^{(k,\ell)} \rangle\right) \geq \min \left\{ f(e) \mid e \in d'^{(k)} \right\} = f'\left(\langle d'^{(k)} \rangle\right) \geq f'(c') = f(c)$. By considering the $\pi$-path $\langle c = c^{(0)} = e^{(0,0)}, \ldots, e^{(0,L_0)} = d^{(0)}; c^{(1)} = e^{(1,0)}, \ldots, e^{(K-1,L_{K-1})} = d^{(K-1)}; c^{(K)} = e^{(K,0)}, \ldots, e^{(K,L_K)} = d^{(K)} = d \rangle$ we see that $d \in C_{(V, \pi, f)}(c)$.

Theorem 1. Let $(V', \pi', f')$ be a digital picture embedded in the digital picture $(V, \pi, f)$, let $C'$ and $D'$ be elements of $C_{(V', \pi', f')}$, and let $c' \in C'$, $d' \in D'$, $c \in c'$ and $d \in d'$ be such that

\[
f(c) = f'(c') = \min \left\{ f'(e') \mid e' \in C' \right\} \quad (3.6)
\]

and

\[
f(d) = f'(d') = \min \left\{ f'(e') \mid e' \in D' \right\} . \quad (3.7)
\]

Then:

1. If $D' = C'$, then $C_{(V, \pi, f)}(c) = C_{(V, \pi, f)}(d)$.
Then the light and dark gray pixels in Figure 3.3(a) — i.e., let of Figures 3.1 and 3.3(b), let shows a digital picture (a) shows a digital picture (V'', \pi'', f'') that is connectedly embedded in the digital picture (V, \pi, f) of Figure 1. (b) shows the partition V'' of V overlaid on the digital picture (V, \pi, f). Elements of V'' comprise 3 \times 3 subarrays of the pixels in V. The components of (V'', \pi'', f'') are V'', P'' = C(V'', \pi'', f'')(p'') = C(V'', \pi'', f'')(s''), Q'' = C(V'', \pi'', f'')(q''), and U'' = C(V'', \pi'', f'')(u''), where P'' consists of the sixteen light gray and dark gray elements of V'', Q'' = \{q''\}, and U'' = \{u''\}.

2. If D' is a proper subset of C', then C(V, \pi, f)(d) is a proper subset of C(V, \pi, f)(c).

Proof. Assertion 1 can be deduced from Lemma 1 by setting D' equal to C' in that lemma. To prove assertion 2, we now assume D' is a proper subset of C' and deduce that C(V, \pi, f)(d) is a proper subset of C(V, \pi, f)(c). It follows from Lemma 1 that C(V, \pi, f)(d) is a subset of C(V, \pi, f)(c). We complete the proof by showing that c \notin C(V, \pi, f)(d). By Proposition 3, (3.6) and (3.7), C' = C(V', \pi', f')(c') and D' = C(V', \pi', f')(d'). Since D' is a subset of C', this implies that f'(c') \leq f'(d'). If it were the case that c \in C(V, \pi, f)(d), then we would have that f'(d') = f(d) \leq f(c) = f'(c') and, consequently, that f'(d') = f'(c') = \min \{f'(e') | e' \in C'\}. Since d' \in C', Proposition 3 would imply that C' = C(V', \pi', f')(d') = D', contradicting the assumption that D' is a proper subset of C'.

To illustrate the first assertion of Theorem 1 we use Figure 3.3. Figure 3.3(a) shows a digital picture (V'', \pi'', f'') that is embedded in the digital picture (V, \pi, f) of Figures 3.1 and 3.3(b). Let P'' be the element of C(V'', \pi'', f'') that comprises all the light and dark gray pixels in Figure 3.3(a) — i.e., let P'' = C(V'', \pi'', f'')(p''). Then p'' and s'' are in P'', p \in p'', s \in s'' and f(p) = f(s) = f''(p'') = f''(s'') =
min \{ f''(e'') | e'' \in P'' \}. According to Theorem 1, we should have that \( C_{(V, \pi, f)}(p) = C_{(V, \pi, f)}(s) \), which is indeed the case.

For an example of the second assertion of Theorem 1, consider the digital pictures in Figures 3.1 and 3.2. \( U' = C_{(V', \pi', f')}(u') = \{ u' \} \) is a proper subset of \( S' = C_{(V', \pi', f')}(s') = \{ s', u' \} \). Since \( s' \in S', u' \in U', s \in s', u \in u', f(s) = f'(s') = \min \{ f'(e') | e' \in S' \} \) and \( f(u) = f'(u') = \min \{ f'(e') | e' \in U' \} \), according to the second assertion of Theorem 1 we should have that \( C_{(V, \pi, f)}(u) \) is a proper subset of \( C_{(V, \pi, f)}(s) \), and this is indeed the case.

The following corollary of Theorem 1 will play an important role in the proof of Theorem 3.

**Corollary 1.** Let \( (V', \pi', f') \) be a digital picture that is embedded in a digital picture \( (V, \pi, f) \) and let \( C', D' \in C_{(V', \pi', f')}, c \in \arg\min_{e \in \bigcup C'} f(e) \) and \( d \in \arg\min_{e \in \bigcup D'} f(e) \). Then:

1. \( C_{(V, \pi, f)}(c) = C_{(V, \pi, f)}(d) \) if \( C' = D' \).

2. \( C_{(V, \pi, f)}(d) \) is a proper subset of \( C_{(V, \pi, f)}(c) \) if \( D' \) is a proper subset of \( C' \).

**Proof.** Let \( e' \) and \( d' \) be the elements of \( C' \) and \( D' \) that contain \( c \) and \( d \), respectively. Then the hypotheses of Theorem 1 are satisfied, and so assertions 1 and 2 follow from the corresponding assertions of the theorem.

\[
\text{3.1.2 Connectedly Embedded Digital Pictures}
\]

A digital picture \( (V', \pi', f') \) is said to be **connectedly embedded in** the digital picture \( (V, \pi, f) \) if \( (V', \pi', f') \) is embedded in \( (V, \pi, f) \) and \( C_{(V, \pi, f)}(c) - \bigcup \{ e' \in V' | f(c) > f'(e') \} \) is \( \pi \)-connected whenever \( c \in \arg\min_{e \in c'} f(e) \) for some \( c' \in V' \).

The digital picture \( (V'', \pi'', f'') \) presented in Figure 3.3(a) is connectedly embedded in the digital picture \( (V, \pi, f) \) presented in Figures 3.1 and 3.3(b). To
Lemma 2. Let \((V', \pi', f')\) be a digital picture connectedly embedded in the digital picture \((V, \pi, f)\), and let \(C\) and \(D\) be elements of \(\mathcal{C}_{(V, \pi, f)}\) such that \(D\) is a subset of
C. Let \( c' \) and \( d' \) be elements of \( V' \) and \( c \) and \( d \) elements of \( V \) such that \( c \in c' \subseteq C \) and \( d \in d' \subseteq D \), and such that

\[
f' (c') = f(c) = \min \{ f(e) \mid e \in C \} \tag{3.8}
\]

and

\[
f' (d') = f(d) = \min \{ f(e) \mid e \in D \}. \tag{3.9}
\]

Then \( C_{(V', \pi', f')}(d') \) is a subset of \( C_{(V, \pi, f)}(c') \).

Proof. It follows from the assumption that \( D \) is a subset of \( C \) and from (3.8) and (3.9) that \( f'(c') = f(c) \leq f(d) = f'(d') \). It follows from Proposition 2 that all we need to show is that \( d' \in C_{(V', \pi', f')}(c') \).

By Proposition 3, (3.8) and (3.9), we have that \( C = C_{(V, \pi, f)}(c) \) and \( D = C_{(V, \pi, f)}(d) \). Let \( B = C_{(V, \pi, f)}(c) \cap \{ e' \in V' \mid f(c) > f'(e') \} \). We now show that both \( c \) and \( d \) are in \( B \). From (3.1), \( c \in C_{(V, \pi, f)}(c) \). On the other hand, \( c \notin \{ e' \in V' \mid f(c) > f'(e') \} \). For otherwise \( c \in e' \in V' \) for some \( e' \) that satisfies \( f(c) > f'(e') \).

But \( c' \) is the unique element in the partition \( V' \) that contains \( c \) and, from (3.8), \( f(c) = f'(c') \). Hence \( c \in B \). From (3.1), \( d \in C_{(V, \pi, f)}(d) = D \subseteq C = C_{(V, \pi, f)}(c) \). On the other hand, \( d \notin \{ e' \in V' \mid f(c) > f'(e') \} \). For otherwise \( d \in e' \in V' \) for some \( e' \) that satisfies \( f(c) > f'(e') \). But \( d' \) is the unique element in the partition \( V' \) that contains \( d \) and we have already shown that \( f(c) \leq f'(d') \). Hence, \( d \in B \).

Since \( (V', \pi', f') \) is connectedly embedded in \( (V, \pi, f) \), \( B \) is \( \pi \)-connected. Hence there exists a \( \pi \)-path \( \langle c^{(0)}, \ldots, c^{(k)} \rangle \) in \( B \) such that \( c^{(0)} = c, c^{(K)} = d \). Since \( B \subseteq C_{(V, \pi, f)}(c), \) it follows from (3.1) that, for 0 \( \leq k \leq K, f(c) \leq f(c^{(k)}) \).

To complete the proof we now construct a \( \pi' \)-path \( \langle d'^{(0)}, \ldots, d'^{(M)} \rangle \) in \( V' \) such that \( d'^{(0)} = c', d'^{(M)} = d' \) and, for 0 \( \leq m \leq M, f'(c') \leq f'(d'^{(m)}). \) Clearly, there exists a strictly increasing sequence of integers 0 \( = k_0, \ldots, k_{M+1} = K + 1 \) such that,
for $0 \leq m \leq M$, \(\{c^{(km)}, \ldots, c^{(km+1-1)}\}\) is a $\pi$-path in an element $d''(m)$ of $V'$ and either $m = M$ or $c^{(km+1)}$ is not an element of $d''(m)$. We now show that \(\langle d''(0), \ldots, d''(M) \rangle\) has the required properties.

By definition $d''(m)$ is in $V'$, for $0 \leq m \leq M$. For $0 \leq m < M$, \(\langle d''(m), d''(m+1) \rangle\) is in $\pi'$. This is indeed so, because $c^{(km+1)}$ is not an element of $d''(m)$ but it is an element of $d''(m+1)$, which implies that $d''(m) \neq d''(m+1)$, and $c^{(km+1-1)}$ is an element of $d''(m)$, such that \(\langle c^{(km+1-1)}, c^{(km+1)} \rangle\) is a $\pi'$-path in $V'$. By definition, $d''(0)$ contains $c^{(k0)} = c^{(0)} = c$. But $c$ is also an element of $c'$, which implies that $d''(0) = c'$. Similarly, $d''(M)$ contains $c^{(kM+1-1)} = c^{(K)} = d$. But $d$ is also an element of $d'$, which implies that $d''(M) = d'$. The only thing left to show is that, for $0 \leq m \leq M$, \(f''(c') \leq f''(d''(m))\). But this is the case, for otherwise it would follow from (3.8) that $f(c') > f''(d''(m))$, which by the definition of $B$ would imply that $c^{(km)}$ is not in $B$, which is a contradiction. \(\square\)

**Theorem 2.** Let \((V', \pi', f')\) be a digital picture connectedly embedded in the digital picture \((V, \pi, f)\) and let $C$ and $D$ be elements of $\mathcal{E}_{(V, \pi, f)}$. Let $c'$ and $d'$ be elements of $V'$ and $c$ and $d$ elements of $V$ such that $c \in c' \subseteq C$ and $d \in d' \subseteq D$, and such that

\[
\begin{align*}
  f''(c') &= f(c) = \min \{f(e) \mid e \in C\} \quad (3.10) \\
  f''(d') &= f(d) = \min \{f(e) \mid e \in D\}. \quad (3.11)
\end{align*}
\]

Then:

1. If $C = D$ then $C_{(V', \pi', f')}(c') = C_{(V', \pi', f')}(d')$.

2. If $D$ is a proper subset of $C$ then $C_{(V', \pi', f')}(d')$ is a proper subset of $C_{(V', \pi', f')}(c')$. 
Proof. Assertion 1 can be deduced from Lemma 2 by setting $D$ equal to $C$ in that lemma. To prove assertion 2, we now assume $D$ is a proper subset of $C$ and deduce that $C(V', \pi', f')(d')$ is a proper subset of $C(V', \pi', f')(c')$. It follows from Lemma 2 that $C(V', \pi', f')(d')$ is a subset of $C(V', \pi', f')(c')$. We complete the proof by showing that $c' \notin C(V', \pi', f')(d')$. By Proposition 3, (3.10) and (3.11), $C = C(V, \pi, f)(c)$ and $D = C(V, \pi, f)(d)$. Since $D$ is a subset of $C$, this implies that $f(c) \leq f(d)$. If it were the case that $c' \in C(V', \pi', f')(d')$, then we would have that $f(d) = f'(d') \leq f'(c') = f(c)$ and, hence, that $f(d) = f(c) = \min \{ f(e) \mid e \in C \}$. Since $d \in C$, Proposition 3 would imply that $C = C(V, \pi, f)(d) = D$, contradicting the assumption that $D$ is a proper subset of $C$.

For examples of Theorem 2, consider the digital pictures presented in Figures 3.1 and 3.3. Referring to Figure 3.1 or 3.3(b), let $S$ denote the component $C(V, \pi, f)(s)$ and let $P$ denote the component $C(V, \pi, f)(p)$. Then $P = S$, and this set consists of all the light gray and dark gray pixels in $V$. In Figure 3.3(a), $s''$ and $p''$ are elements of $V''$, and we have that $s \in s'' \subseteq S$, $p \in p'' \subseteq P$, $f''(p'') = f(s) = \min \{ f(e) \mid e \in S \}$ and $f''(p'') = f(p) = \min \{ f(e) \mid e \in P \}$. Since $P = S$, according to the Theorem 2 we should have that $C(V', \pi', f')(p'') = C(V', \pi', f')(s'')$, which is indeed the case. Referring again to Figures 3.1 and 3.3(b), let $U$ denote the component $C(V, \pi, f)(u)$, which consists of the twenty dark gray pixels on the right side of $V$ and is a proper subset of $S$. In Figure 3.3(a), $u''$ is an element of $V''$, and we have that $u \in u'' \subseteq U$ and $f''(u'') = f(u) = \min \{ f(e) \mid e \in U \}$. Since $U$ is a proper subset of $S$, according to Theorem 2 we should have that $C(V', \pi', f')(u'')$ is a proper subset of $C(V', \pi', f')(s'')$, and this is indeed so.

The following corollary of Theorem 2 is a partial converse of Corollary 1:

**Corollary 2.** Let $(V', \pi', f')$ be a digital picture that is connectedly embedded in a digital picture $(V, \pi, f)$ and let $C', D' \in \mathcal{C}(V', \pi', f')$, $c \in \arg\min_{e \in \bigcup C'} f(e)$ and $d \in
argmin_{e \in \bigcup D'} f(e). Then:

1. $C' = D'$ if $C_{(V, \pi, f)}(c) = C_{(V, \pi, f)}(d)$.

2. $D'$ is a proper subset of $C'$ if $C_{(V, \pi, f)}(d)$ is a proper subset of $C_{(V, \pi, f)}(c)$.

Proof. Let $c'$ and $d'$ be the elements of $C'$ and $D'$ that contain $c$ and $d$, respectively. Since $c \in \arg\min_{e \in \bigcup C'} f(e)$ and $c \in c' \subseteq \bigcup C'$, we have that $c \in \arg\min_{e \in c'} f(e)$. Hence (recalling Remark 1) $c' \subseteq C_{(V, \pi, f)}(c)$.

Moreover, $f'(c') = f(c) = \min \{f(e) \mid e \in \bigcup C'\} = \min \{f'(e') \mid e' \in C'\}$, and so $C' = C_{(V', \pi', f')}(c')$ (by Proposition 3). Analogously, $d' \subseteq C_{(V, \pi, f)}(d)$, $f'(d') = f(d) = \min \{f(e) \mid e \in \bigcup D'\} = \min \{f'(e') \mid e' \in D'\}$, and $D' = C_{(V', \pi', f')}(d')$. Putting $C = C_{(V, \pi, f)}(c)$ and $D = C_{(V, \pi, f)}(d)$, the hypotheses of Theorem 2 are satisfied and so assertions 1 and 2 follow from the corresponding assertions of the theorem.

3.1.3 The Component Tree of a Digital Picture

Recall that a rooted tree $T$ is a pair $(N, E)$, where $N$ is a finite set of nodes and $E$ is a set of edges. Each edge is an ordered pair of distinct nodes that are respectively called the parent node and the child node of the edge, and the nodes and edges satisfy the following conditions: (1) every member of $N$, except one element called the root, is a child node of just one edge; (2) the reflexive transitive closure of $E$ is a partial order on $N$. If $m$ and $n$ are nodes such that $m = n$ or $m$ precedes $n$ in the partial order, then $n$ is called a descendant of $m$ and $m$ is called an ancestor of $n$. In particular, every node in $N$ is a descendant of the root. We say $m$ is a proper descendant (respectively, proper ancestor) of $n$ if $m$ is a descendant (respectively, an ancestor) of $n$ and $m \neq n$.

For any digital picture $(V, \pi, f)$, the tree $T_{(V, \pi, f)} = (N_{(V, \pi, f)}, E_{(V, \pi, f)})$ of $(V, \pi, f)$ is the rooted tree for which

1. the set of nodes $N_{(V, \pi, f)}$ is $\mathcal{C}_{(V, \pi, f)}$, 

2. the root is $V$, and

3. for any $C$ and $D$ in $N(V, \pi, f) = \mathcal{C}(V, \pi, f)$, $C$ is an ancestor of $D$ if, and only if, $D$ is a subset of $C$.

That these conditions define a tree follows from an observation we made after defining $\mathcal{C}(V, \pi, f)$: Any two members of $\mathcal{C}(V, \pi, f)$ are disjoint unless one is a subset of the other. This observation implies that if $D \in \mathcal{C}(V, \pi, f)$ is a subset both of $C_1 \in \mathcal{C}(V, \pi, f)$ and of $C_2 \in \mathcal{C}(V, \pi, f)$ then one $C_1$ and $C_2$ must be a subset of the other. So for each $D \in \mathcal{C}(V, \pi, f) - \{V\}$ there exists a unique smallest member $B$ of $\mathcal{C}(V, \pi, f)$ that contains $D$ as proper subset, and $(B, D)$ will be the unique edge in $E(V, \pi, f)$ that has $D$ as its child node.

This tree is often called the *component tree* [20, 63, 21], and is very similar to the *foreground history tree* of [64]. It is isomorphic to the *join tree* of [65] if $f$ is 1-to-1.

### 3.1.3.1 Component Tree of a 1-Dimensional Digital Picture

As a first and very simple illustration of how a component tree can be created from a digital picture, we explain the construction of the component tree of the 1-dimensional digital picture shown in Figure 3.4.

This digital picture $(V, \pi, f)$ contains just a single row of 37 elements, each element is represented by a small square in Figure 3.4. The intensity of each element is indicated by the number above that element. For example, the intensities of the four leftmost elements are respectively 0, 3, 14, and 14. The $\pi$-adjacency relation
Figure 3.5: Detailed and Simplified Representations of a Component Tree.
(a) Component tree of the digital picture \((V, \pi, f)\) of Figure 3.4, shown in full
detail—for each node, the node’s level and the elements that constitute that node
are shown. (b) Simplified drawing of the same tree in which each node is shown
just as a point.

of the digital picture is edge adjacency—two elements are considered to be adjacent
if they share an edge. Thus each element, except for the first and the last element
in the row, is adjacent to just two elements: one on its left and one on its right. The
first and the last elements are each adjacent to just one element.

The component tree \(T_{(V, \pi, f)}\) of \((V, \pi, f)\) is shown in Figure 3.5(a). Note that
\((V, \pi, f)\) is reproduced at the top of Figure 3.5(a); this is to make it easier to verify
certain relationships between the tree and the picture that we will state below.

Each node of the component tree \(T_{(V, \pi, f)}\) is a set of elements of \(V\); the cardinal-
ity of a node is the number of elements in that set. For example, we see from Figure 3.5(a) that the node $v_0$ has cardinality 37—it is just the set of all 37 elements. We also see that the node $v_1$ has cardinality 36 (as it is the set of all elements other than the leftmost element), and that the node $v_{20}$ has cardinality two (as it consists of the 2nd and the 3rd elements from the right).

However, we often draw component trees more simply, by showing each node as a point rather than a set of elements. Figure 3.5(b) shows the same component tree in this simplified way.

Every node of the component tree $T_{(V,\pi,f)} = (N_{(V,\pi,f)}, E_{(V,\pi,f)})$ of $(V,\pi,f)$ has a level; the level of any node $C \in N_{(V,\pi,f)}$ is defined to be the minimum of the intensity levels of the elements of $C$. In Figure 3.5(a), the levels of the nodes are indicated by the numbers beside the vertical bar on the left. For example, we see at once that the level of the node $v_9$ is 10. (It is also easy to verify that this is correct: The node $v_9$ consists of eight elements whose intensity levels are 14, 14, 12, 14, 14, 10, 12, and 12, and the minimum of these intensity levels is indeed 10.)

We now describe an easy way to create the component tree $T_{(V,\pi,f)}$ from a digital picture $(V,\pi,f)$. This will involve thresholding the digital picture $(V,\pi,f)$ at every distinct intensity levels of the elements of $V$. For any intensity level $t$, we threshold a digital picture $(V,\pi,f)$ at the level $t$ by omitting all the $c \in V$ such that $f(c) < t$. This process will create a $t$-superlevel set $V_{f,t}$ of $(V,\pi,f)$. Each maximal connected fragment of $V_{f,t}$ is a $\pi$-component components of $V_{f,t}$; if $V_{f,t}$ is disconnected, then it will consist of two or more $\pi$-connected components. For example, if we threshold the picture of Figure 3.4 at level $t = 16$, then just six elements will be retained—the 2nd, 3rd, 5th, 6th, 8th, and 9th elements from the right—and those six elements will fall into three components, each of which consists of just two adjacent elements.

As mentioned above, the direct way of constructing a component tree that will
be described below involves thresholding the digital picture at every intensity level that occurs in the picture. Another way of constructing component trees is presented in Najman and Couprie 2006 [63]. The algorithm of [63] does not involve thresholding and is computationally more efficient when applied to digital pictures that have many intensity levels. It processes the picture elements in decreasing order of their intensity and uses Tarjan’s union-find algorithm [66] to build the tree from the bottom up. The algorithm used to compute the component tree in the context of this dissertation was first presented in [64]. The time-complexity to compute a component tree for a digital picture \((V, \pi, f)\) is \(O(|V| \log |M|)\) where \(M\) is the set of all the intensity levels that occur in \((V, \pi, f)\).

How We Can Find the Nodes of the Component Tree

Let \((V, \pi, f)\) be the digital picture, and let \(M\) be the set of all the intensity levels that occur in \((V, \pi, f)\). The nodes of the component tree of the digital picture \((V, \pi, f)\) can be found by thresholding \((V, \pi, f)\) at each of the graylevels in \(M\). At each threshold level \(t \in M\), we find the picture elements whose intensity levels are \(\geq t\) and then find the connected components of that set of picture elements. Each such connected component is one node of the component tree. As stated above, we define the level of that node to be the minimum of the intensity levels of the picture elements in the component.

Every node of the component tree can be obtained in this way. However, the level of a node that is found when \((V, \pi, f)\) is thresholded at intensity level \(t\) need not be \(t\): Such a node may have level \(t' > t\), in which case that very same node will also be found when \((V, \pi, f)\) is thresholded at any other intensity level between \(t\) and \(t'\).
Examples of How Nodes are Found

Consider in Figure 3.4 the threshold level $t = 0$. In this case, all 37 elements of $(V, \pi, f)$ have intensity level $\geq t = 0$, so the set of elements with intensity level $\geq t = 0$ has just one $\pi$-connected component (namely the entire set of 37 elements). Thus thresholding the picture $(V, \pi, f)$ at level $t = 0$ yields just one node of the component tree $T_{(V, \pi, f)}$. This node is shown as $v_0$ in Figure 3.5(a). The node’s level (i.e., the minimum of the intensities of its elements) is 0 because the intensity of the leftmost element of $(V, \pi, f)$ is 0. This node is the root node of the component tree $T_{(V, \pi, f)}$.

Now let us consider the threshold level $t = 1$. In this case, all but one of the 37 elements of $(V, \pi, f)$ have intensity levels $\geq t = 1$; the only exception is the leftmost element, whose intensity is 0. This set of 36 elements also has only one $\pi$-component. Thus thresholding the picture $(V, \pi, f)$ at level 1 yields just one node of the component tree. This node (of cardinality 36) is shown as $v_1$ in Figure 3.5(a). The minimum of the intensities of the elements in this node is 1 (because the intensity of the 18th element is 1), so this node has level 1.

Thresholding the picture $(V, \pi, f)$ at the next level in $\ell$, namely the level $t = 3$, yields two nodes of the component tree that have cardinalities 16 and 19. This is because the set of elements with intensity $\geq 3$ consists of the two components labeled $v_2$ and $v_3$ in Figure 3.5(a), which are separated by an element whose intensity is 1. In each of the two components the element of lowest intensity has intensity 3, so each of the two nodes has level 3.

The next threshold level in $M$ is $t = 6$. The reader should now have no difficulty in verifying that thresholding $(V, \pi, f)$ at level $t = 6$ yields just two nodes of the tree, both of which have level 6. These nodes, which have cardinalities 15 and 18, are labeled $v_4$ and $v_5$ in Figure 3.5(a).

For the threshold levels $t$ that have been considered so far, the component tree
nodes that are found when we threshold \((V, \pi, f)\) at intensity level \(t\) have also had level \(t\). But this is not true when we use the threshold level \(t = 7\). The threshold level \(t = 7\) will yield five component tree nodes because the set of elements with intensity \(\geq 7\) consists of five components. But only one of these five nodes will have a level that is equal to \(t\); the levels of the other four nodes will be higher.

Indeed, the leftmost component of the set of elements with intensity \(\geq 7\), labeled \(v_9\) in Figure 3.5(a), consists of eight elements with intensities 14, 14, 12, 14, 14, 10, 12, and 12; this will therefore be a component tree node whose level is 10. Another component, labeled \(v_7\), consists of five elements with intensities 14, 14, 8, 9, and 10; this will therefore be a component tree node whose level is 8. A third component, labeled \(v_6\), consists of four elements with intensities 7, 7, 12, and 12; this will be a node whose level is 7. A fourth component, labeled \(v_{14}\), consists of two elements that both have intensity 12; this will be a node whose level is 12. The fifth component, labeled \(v_{15}\), consists of eight elements with intensities 18, 18, 13, 18, 18, 12, 16, and 18; this too will be a node whose level is 12.

**How We Can Find the Edges of the Component Tree**

The edges of the component tree connect nodes at different levels in a way that reflects the inclusion relationships between nodes. Specifically, there is an edge from a node \(C\) to a node \(D\) of higher level just if \(C\) is the node of highest level such that \(C \supset D\) (i.e., just if the set of picture elements \(C\) strictly contains the set of picture elements \(D\) and there is no node of higher level than \(C\) that strictly contains \(D\)).

It is easily verified that the edges shown in Figure 3.5(a) are exactly the edges given by this rule. For example, we see from Figure 3.5 that \(v_{20}\) is the node of highest level that strictly contains the node \(v_{23}\) and so there is an edge from \(v_{20}\) to \(v_{23}\) in the tree.
Figure 3.6: A Simple 2-Dimensional Digital Picture.

(a) Surface rendering of a three-dimensional reconstruction (from cryo-electron microscopy images) of helicase DnaB (EMDB access code 1022 [67]). (b) The central slice of the density map. (c) The reduced version of (b) that is used in Sub-subsection 3.1.3.2: In (c), each element corresponds to a $5 \times 5$ region of the image in (b) and element intensity levels have been quantized to five values 0, 1, 2, 3, and 4 (which are respectively shown as black, dark gray, gray, light gray, and white).

3.1.3.2 Component Tree of a 2-Dimensional Digital Picture

The above process of creating a component tree is valid for digital pictures of any dimension. An additional example of how component trees are constructed is presented in this sub-subsection. This time, a digital picture was created based on a real biological structure. Figure 3.6 show how this 2D digital picture was created from a slice of a density map.

The digital picture in Figure 3.6(c) is a simplified version of a central slice of a three-dimensional reconstruction (from cryo-electron microscopy images) of helicase DnaB (EMDB access code 1022 [67]). Figure 3.6(a) shows a surface rendering of the three-dimensional density map. Figure 3.6(b) shows a cropped part of the original central slice, which contains $50 \times 50$ elements. This was simplified to Figure 3.6(c), which contains $10 \times 10$ elements, by replacing $5 \times 5$ arrays of elements by single elements, whose intensities were obtained by averaging the intensities of the elements in the corresponding arrays, and by quantizing element intensity levels to a set of five equally spaced values represented by the integers $0, \ldots, 4$.
Figure 3.7: Components of the Digital Picture of Figure 3.6(c).
(a) Component tree of the 2D digital picture presented in Figure 3.6(c), shown using the simplified representation in which each node appears as a point rather than as a set of elements. (b), (c), (d) and (e) The components at threshold levels 1, 2, 3, and 4, respectively; in each case the cross-hatched parts of the image consist of elements that do not belong to any component because their intensities are below the threshold level. Each component shown in (b), (c), (d) and (e) is a node of the tree (a): Tree node $v_1$ consists of the 98 elements that are not cross-hatched in (b), and tree nodes $v_2$, ..., $v_8$ are the correspondingly labeled components in (c), (d) and (e).

3.6(c), the intensity levels 0, 1, 2, 3, and 4 are respectively shown as black, dark gray, gray, light gray, and white. We regard Figure 3.6(c) as a digital picture in which the adjacency relation within the set of elements is edge adjacency: Distinct elements are considered to be adjacent just if they share an edge.

The component tree of the digital picture 3.6(c) is shown in Figure 3.7 using the simplified representation in which each node is shown as a point rather than as a set of elements. We will now describe a construction of this tree.

When the picture is thresholded at the lowest intensity level $t = 0$, there is just one component, which consists of all 100 elements in the image since all elements have intensity $\geq t = 0$. Thus thresholding the picture at level $t = 0$ yields just this one node, which is the root $v_0$ of the component tree in Figure 3.7(a). The node’s
level (i.e., the minimum of the intensities of its elements) is 0.

When the digital \((V, \pi, f)\) presented in Figure 3.6(c) is thresholded at the intensity level \(t = 1\), there is again just one component, because the two elements of the image that have intensity less than 1 (the two black elements in Figure 3.6(c), which are cross-hatched in Figure 3.7(b) do not separate \(v_0\). This component is the tree node \(v_1\) in Figure 3.7(a); its cardinality is 98, and its level is 1 because it does contain elements whose intensity is 1.

When the picture \((V, \pi, f)\) presented in Figure 3.6(c) is thresholded at the intensity level \(t = 2\), all elements in the cross-hatched parts of Figure 3.7(c) have intensity levels that are below the threshold and are therefore omitted. The remaining elements belong to two components: As we see from 3.7(c), one component consists of a single element in the top left of the image (node \(v_2\) in the tree) and the second component consists of all the other elements with intensity \(\geq t = 2\) (node \(v_3\) in the tree). Since the only element in \(v_2\) has intensity 2, and there are many elements in \(v_3\) that have intensity 2, both of these nodes have level 2.

When the picture \((V, \pi, f)\) is thresholded at the intensity level \(t = 3\), there are again two components: All elements in the cross-hatched parts of Figure 3.7(d) have intensity levels that are below the threshold \(t = 3\); the remaining elements consist of a component \(v_4\) of cardinality 5 and a component \(v_5\) of cardinality 12. We also see from Figure 3.7(d) that each of \(v_4\) and \(v_5\) contains elements that have intensity 3—\(v_4\) has two such elements and \(v_5\) has three—so each of \(v_4\) and \(v_5\) is a node of level 3 in the component tree. These level 3 nodes are children of the level 2 node \(v_3\) because each of the sets \(v_4\) and \(v_5\) is contained in the set \(v_3\).

When the picture \((V, \pi, f)\) is thresholded at the intensity level 4, there are three components: All elements in the cross-hatched parts of Figure 3.7(e) have intensity levels that are below the threshold, and the remaining elements consist of a component \(v_6\) of cardinality 3, a component \(v_7\) of cardinality 4, and a component \(v_8\) of
Figure 3.8: Component Trees of Digital Pictures.

(a) is a copy of Figure 3.2(b); it shows the digital picture \((V, \pi, f)\) of Figure 3.1 overlaid on the partition \(V'\) of \(V\) that is shown in Figure 3.2(a). (c) shows the rooted tree \(T_{(V, \pi, f)}\) of \((V, \pi, f)\). (b) is a copy of Figure 3.2(a); it shows a digital picture \((V', \pi', f')\) that is embedded in the digital picture \((V, \pi, f)\). (d) shows the rooted tree \(T_{(V', \pi', f')}\) of \((V', \pi', f')\). In our drawings of rooted trees, each edge of the tree is represented by a line segment whose upper and lower endpoints represent the parent node and the child node of the edge. In the rooted tree \(T_{(V, \pi, f)}\), 
\[N_{(V, \pi, f)} = \{V, P, Q, R, U\}\]
where 
\[P = C_{(V, \pi, f)}(p)\]
comprises all the light gray and dark gray elements in (a), 
\[Q = C_{(V, \pi, f)}(q)\]
comprises the nine dark gray elements on the left side of (a), and 
\[R = C_{(V, \pi, f)}(r) = \{r\}\].
In the rooted tree \(T_{(V', \pi', f')}\), 
\[N_{(V', \pi', f')} = \{V', P', S', U'\}\]
where 
\[P' = C_{(V', \pi', f')}(p')\]
comprises the four light gray elements of \(V'\) on the left side of (b), 
\[S' = C_{(V', \pi', f')}(s') = \{s', u'\}\], and 
\[U' = C_{(V', \pi', f')}(u') = \{u'\}\].

Cardinality 5. All the elements in these components have intensity 4, so each of \(v_6\), \(v_7\), and \(v_8\) is a node of level 4 in the component tree. From 3.7(d) and 3.7(e) we see that \(v_6\) is contained in the level 3 node \(v_4\) and must therefore be a child of \(v_4\) in the tree. We similarly see that \(v_7\) and \(v_8\) are both contained in the level 3 node \(v_5\) and must therefore be children of that node.

As additional examples, we present the component trees of the digital pictures in Figure 3.2. Figure 3.8 shows the digital pictures \((V, \pi, f)\) and \((V', \pi', f')\) of Figures 3.2(b) and 3.2(a) and their respective rooted trees \(T_{(V, \pi, f)}\) and \(T_{(V', \pi', f')}\).
Figure 3.9: Embedding of Component Trees.
(a) is a copy of Figure 3.3(b); it shows the digital picture \((V, \pi, f)\) of Figure 3.1 overlaid on the partition \(V''\) of \(V\) that is shown in Figure 3.3(a). (b) is a copy of Figure 3.3(a); it shows a digital picture \((V''', \pi''', f''')\) that is embedded in the digital picture \((V, \pi, f)\). The red arrows in (c) show a tree embedding of the tree \(T_{(V''', \pi''', f''')}\) into the tree \(T_{(V, \pi, f)}\). The non-root nodes \(P, Q, R,\) and \(U\) of \(T_{(V, \pi, f)}\) were specified in the caption of Figure 3.8. In \(T_{(V''', \pi''', f''')}\) the non-root nodes are \(P''' = C_{(V''', \pi''', f''')}(p''')\) (which is the set of all the light gray and dark elements in \(V''\)), \(Q''' = C_{(V''', \pi''', f''')}(q''') = \{q''\}\), and \(U''' = C_{(V''', \pi''', f''')}(u''') = \{u''\}\).

3.1.4 Tree Embeddings

Recall that a tree embedding of a rooted tree \(T' = (N', E')\) into a rooted tree \(T = (N, E)\) is a mapping \(\varphi_e : N' \to N\) such that \(C'\) is a descendant of \(D'\) in \(T'\) if, and only if, \(\varphi_e(C')\) is a descendant of \(\varphi_e(D')\) in \(T\). Note that every tree embedding \(\varphi_e\) is a 1-to-1 mapping, because if \(\varphi_e(C') = \varphi_e(D')\) then each of \(\varphi_e(C')\) and \(\varphi_e(D')\) is a descendant of the other, whence each of \(C'\) and \(D'\) is a descendant of the other, which implies \(C' = D'\). Tree embedding has been defined in various slightly different ways in the literature. Our definition coincides with what is called minor embedding in [55]; it is illustrated in Figure 3.9. The notion of tree embedding has been used in numerous applications of combinatorial computer science [54, 56, 57, 59].
The tree embedding shown in Figure 3.9 is an example of what we call a natural component embedding, a concept we now define. Let \((V', \pi', f')\) be a digital picture that is embedded in a digital picture \((V, \pi, f)\). We define the natural component embedding associated with \((V', \pi', f')\) and \((V, \pi, f)\) to be the mapping \(\phi : \mathcal{C}(V', \pi', f') \rightarrow \mathcal{C}(V, \pi, f)\) such that if \(C' \in \mathcal{C}(V', \pi', f')\) then \(\phi(C') = C(V, \pi, f)(e)\), where \(e\) is an arbitrary element of \(\text{argmin}_{e \in \bigcup C' f(e)}\). There is exactly one mapping \(\phi : \mathcal{C}(V', \pi', f') \rightarrow \mathcal{C}(V, \pi, f)\) that satisfies this condition.

For example, the natural component embedding associated with the digital pictures \((V', \pi', f')\) and \((V, \pi, f)\) in Figure 3.8 is the map \(\phi : \mathcal{C}(V', \pi', f') \rightarrow \mathcal{C}(V, \pi, f)\) defined by \(\phi(V') = V\), \(\phi(P') = \phi(S') = P\), and \(\phi(U') = U\). Here it can be observed that:

- \(\phi(V') = V\) because for any \(e \in \text{argmin}_{e \in V} f(e) = \text{argmin}_{e \in V} f(e)\), we have that \(C(V, \pi, f)(e) = V\);
- \(\phi(P') = P\) because \(p \in \text{argmin}_{p \in P} f(e)\) and \(C(V, \pi, f)(p) = P\);
- \(\phi(S') = P\) because \(s \in \text{argmin}_{s \in S} f(e)\) and \(C(V, \pi, f)(s) = P\);
- \(\phi(U') = U\) because \(u \in \text{argmin}_{u \in U} f(e)\) and \(C(V, \pi, f)(u) = U\).

It is easily verified that the tree embedding shown in Figure 3.9 of the tree \(T(V'', \pi'', f'')\) into the tree \(T(V, \pi, f)\) is a natural component embedding, as we mentioned above. Note that whereas this natural component embedding is a tree embedding, the natural component embedding \(\phi : \mathcal{C}(V', \pi', f') \rightarrow \mathcal{C}(V, \pi, f)\) discussed in the previous paragraph was not—indeed, that map \(\phi\) was not even 1-to-1. We will see from the main result of this section, Theorem 3, that this difference between the two natural component embeddings reflects the fact that the digital picture \((V'', \pi'', f'')\) of Figure 3.9 is connectedly embedded in the digital picture \((V, \pi, f)\) whereas the digital picture \((V', \pi', f')\) of Figure 3.8 is not.

The following lemma will be used in the proof of Theorem 3.
Lemma 3. Let \((V', \pi', f')\) be a digital picture embedded in the digital picture \((V, \pi, f)\), let \(\phi : \mathcal{C}(V', \pi', f') \to \mathcal{C}(V, \pi, f)\) be the natural component embedding associated with this pair of pictures and let \(C' \subseteq \mathcal{C}(V', \pi', f')\), \(c' \in C'\) and \(c \in c'\) be such that
\[
f(c) = f'(c') = \min \{ f'(c') | c' \in C' \}
\] (3.12)
Then \(\bigcup C'\) is a \(\pi\)-connected subset of \(\phi(C') = C(V, \pi, f)(c)\).

Proof. In view of Remark 1 and (3.12), we need only show that \(\bigcup C'\) is \(\pi\)-connected. Let \(a\) and \(b\) be any two elements of \(\bigcup C'\), and let the elements of \(C'\) that contain \(a\) and \(b\) be \(d'\) and \(b'\), respectively. As \(C'\) is \(\pi'\)-connected, there is a \(\pi'\)-path \(\langle d'^{(0)}, \ldots, d'^{(K)} \rangle\) in \(C'\) such that \(d'^{(0)} = a'\) and \(d'^{(K)} = b'\). By the definition of \(\pi'\), for \(0 \leq k < K\) there exist \(d^{(k)} \in d'^{(k)}\) and \(c^{(k+1)} \in d'^{(k+1)}\) such that \((d^{(k)}, c^{(k+1)}) \in \pi\).
We also define \(e^{(0)} = a\) and \(d^{(K)} = b\). With these definitions it is the case that, for \(0 \leq k \leq K\), both \(e^{(k)}\) and \(d^{(k)}\) are in \(d'^{(k)} \in V'\). By the definition of an embedded digital space, \(d'^{(k)}\) is a \(\pi\)-connected subset of \(V\) and so there exists a \(\pi\)-path \(\langle e^{(k,0)}, \ldots, e^{(k,L_k)} \rangle\) in \(d'^{(k)}\) such that \(e^{(k,0)} = e^{(k)}\) and \(e^{(k,L_k)} = d^{(k)}\). By considering the \(\pi\)-path \(\langle a = e^{(0)} = e^{(0,0)}, \ldots, e^{(0,L_0)} = d^{(0)}, c^{(1)} = e^{(1,0)}, \ldots, e^{(K-1,L_{K-1})} = d^{(K-1)}, c^{(K)} = e^{(K,0)}, \ldots, e^{(K,L_K)} = d^{(K)} = b \rangle\) we see that \(a\) is \(\pi\)-connected to \(b\) in \(\bigcup C'\). Since \(a\) and \(b\) are arbitrary elements of \(\bigcup C'\), we deduce that \(\bigcup C'\) is \(\pi\)-connected, as required.

Theorem 3. Let \((V', \pi', f')\) be a digital picture embedded in the digital picture \((V, \pi, f)\) and let \(\phi : \mathcal{C}(V', \pi', f') \to \mathcal{C}(V, \pi, f)\) be the associated natural component embedding. Then \(\phi\) is a tree embedding of \(T_{(V', \pi', f')}\) into \(T_{(V, \pi, f)}\) if, and only if, \((V', \pi', f')\) is connectedly embedded in \((V, \pi, f)\).

Proof. Let us first establish that if the digital picture \((V', \pi', f')\) is connectedly embedded in \((V, \pi, f)\), then \(\phi\) is a tree embedding of \(T_{(V', \pi', f')}\) into \(T_{(V, \pi, f)}\). What we
need to establish is that if \((V', \pi', f')\) is connectedly embedded in \((V, \pi, f)\), then \(D'\) is a descendant of \(C'\) in \(T_{(V', \pi', f')}\) if, and only if, \(\phi(D')\) is a descendant of \(\phi(C')\) in \(T_{(V, \pi, f)}\). Readily, the “if” part follows from Corollary 2 (and the definition of \(\phi\)) and the “only if” part follows from Corollary 1. To prove the converse, we now suppose \(\phi\) is a tree embedding of \(T_{(V', \pi', f')}\) into \(T_{(V, \pi, f)}\), and deduce that \((V', \pi', f')\) is connectedly embedded in \((V, \pi, f)\). Since \((V', \pi', f')\) is embedded in \((V, \pi, f)\), we only need show that \(C_{(V, \pi, f)}(c) - \bigcup\{e' \in V' \mid f(c) > f'(e')\}\) is \(\pi\)-connected whenever \(c \in \text{argmin}_{e \in e'} f(e)\) for some \(e' \in V'\). So let \(c'\) be any element of \(V'\), let \(c \in \text{argmin}_{e \in e'} f(e)\) (so that \(f'(c') = f(c)\)), and let \(B = C_{(V, \pi, f)}(c) - \bigcup\{e' \in V' \mid f(c) > f'(e')\}\). If we can show \(B\) is \(\pi\)-connected, then the theorem will be proved.

Let \(C' = C_{(V', \pi', f')} (c')\), so that (3.12) holds. Then \(c' \in \text{argmin}_{e \in C'} f'(e')\) and therefore \(c \in \text{argmin}_{e \in \bigcup C'} f(e)\). We now show that \(\bigcup C' = B\); this will imply that \(B\) is \(\pi\)-connected (in view of Lemma 3), as required.

Suppose \(\bigcup C' \neq B\). By (3.12), the subset \(C'\) of \(V'\) satisfies \(C' \cap \{e' \in V' \mid f(c) > f'(e')\} = \emptyset\). So, since distinct members of \(V'\) are disjoint, we have that \(\bigcup C' \cap \bigcup\{e' \in V' \mid f(c) > f'(e')\} = \emptyset\). We also have that \(\bigcup C' \subseteq C_{(V, \pi, f)}(c)\), by Lemma 3. Hence \(\bigcup C' \subseteq B\) and so our assumption that \(\bigcup C' \neq B\) implies there is some \(d \in B\) such that \(d \notin \bigcup C'\). Let \(d'\) be the element of \(V'\) such that \(d \in d'\), and let \(D' = C_{(V', \pi', f')} (d')\). Then \(D' \notin C'\), as \(d \in d' \in D'\) but \(d \notin \bigcup C'\). But we will now show that \(\phi(D') \subseteq \phi(C')\). This contradiction of the hypothesis that \(\phi\) is a tree embedding will show that \(\bigcup C' = B\) is impossible. As we observed earlier, that will be enough to prove the theorem.

To show that \(\phi(D') \subseteq \phi(C')\), let \(a\) be an element of \(\text{argmin}_{e \in d'} f(e)\), so that \(f'(d') = f(a)\). As \(d' \in \text{argmin}_{e \in D'} f'(e')\), we have that \(a \in \text{argmin}_{e \in \bigcup D'} f(e)\). As \(a \in \text{argmin}_{e \in \bigcup D'} f(e)\) and \(c \in \text{argmin}_{e \in \bigcup C'} f(e)\), we see from the definition of \(\phi\) that \(\phi(D') = C_{(V, \pi, f)} (a)\) and \(\phi(C') = C_{(V, \pi, f)} (c)\). So it remains only to show that \(C_{(V, \pi, f)} (a) \subseteq C_{(V, \pi, f)} (c)\).
Figure 3.10: Illustration of Potential Applicability of Embedded Digital Pictures. (a) A 50 × 50 pixel digital picture obtained from slice 15 of a density map with EMDB access code 1022 [67] in the EMDataBank (see text). (b) A 25 × 25 pixel digital picture that is connectedly embedded in the digital picture in (a). (c) A 50 × 50 pixel digital picture obtained from slice 23 of the entry EMD-1022 in the EMDataBank. (d) A 25 × 25 pixel digital picture that is connectedly embedded in the digital picture in (c).

Note that $d' \notin \{e' \in V' \mid f(c) > f'(e')\}$ (since $d \in d'$, and $d \notin \bigcup \{e' \in V' \mid f(c) > f'(e')\}$ because $d \in B$). So $f(c) \leq f'(d') = f(a)$. By Remark 1, $d' \subseteq C_{(V, \pi, f)}(a)$. As $d \in d' \subseteq C_{(V, \pi, f)}(a)$, there is a $\pi$-path in $V_{f,f(a)}$ connecting $a$ to $d$. As $d \in B \subseteq C_{(V, \pi, f)}(c)$, there is a $\pi$-path in $V_{f,f(c)}$ connecting $c$ to $d$. Since $f(c) \leq f(a)$, we have that $V_{f,f(a)} \subseteq V_{f,f(c)}$. Thus when we concatenate the latter $\pi$-path and the reverse of the former $\pi$-path we obtain a $\pi$-path in $V_{f,f(c)}$ connecting $c$ to $a$. Hence $a \in C_{(V, \pi, f)}(c)$. It follows (by Proposition 2) that $C_{(V, \pi, f)}(a) \subseteq C_{(V, \pi, f)}(c)$, and so our proof is complete.

3.1.5 Illustration of Potential Applicability in Structural Biology

To illustrate the potential applicability of our theory to structural biology, two digital pictures were produced from two slices of a density map with EMDB ID 1022 [67]. The density map consist of 50 × 50 × 50 voxels representing a DnaB from E.coli and was obtained by cryo-EM. These digital pictures are represented in Figures 3.10(a) and (c). Just as discussed in the caption of Figure 3.1, such images represent a digital picture $(V, \pi, f)$, where $V$ is the set of pixels, $\pi$ is edge-adjacency and $f(p)$
Figure 3.11: Illustration of Potential Applicability of Embedded Component Trees. The component trees from the digital pictures presented in Figure 3.10(a), (b), (c) and (d) are presented in (a), (b), (c) and (c), respectively.

is the gray level of the pixel $p$ in the image. Similarly to what was done in Figures 3.2 and 3.3, for each of these $50 \times 50$ pixel digital pictures $(V, \pi, f)$, we created a digital picture $(V', \pi', f')$ in which each element of $V'$ is a $2 \times 2$ subarray of the pixels in $V$. The two $(V', \pi', f')$s are represented in Figures 3.10(b) and (d). These representations also have an alternative interpretation as $25 \times 25$ digital pictures $(V'', \pi'', f'')$, where the elements of $V''$ are considered to be pixels in the $25 \times 25$ digital pictures rather than sets of four pixels in the $50 \times 50$ pixel digital pictures.

For each of these four digital pictures we also created their corresponding component trees. These are shown in Figure 3.11. The representation of the trees in that figure needs some explanation. There are four kinds of nodes in our rooted trees: (1) The root, which is represented by the upper endpoint of the uppermost vertical line. (2) Nodes that are not the parent node of any edge (usually referred
to as leaves); whenever a vertical line’s lower endpoint does not lie on a horizontal line, that lower endpoint represents a leaf. (3) Nodes that are the parent nodes of more than one edge, which are represented by horizontal lines. (4) Nodes other than the root that are the parent node of exactly one edge, which are not explicitly represented but correspond to points that occur somewhere in the interior of a vertical line.

We now use the trees of Figure 3.11 to show how the trees associated with digital pictures can be used to answer questions about those digital pictures. Recall that Figures 3.10(a) and (b) show two images of the same biological object at different resolutions, as do Figures 3.10(c) and (d). But now suppose we are presented with the two digital pictures \((V, \pi, f)\) of Figure 3.10(a) and \((V'', \pi'', f'')\) of Figure 3.10(d) and we are interested in the possibility that they are images of the same biological object obtained at different resolutions.

Let \(T\) be a rooted tree. A node \(C\) of \(T\) is said to be critical if the number of children of \(T\) is different from one. In our tree representation explained above, critical nodes are represented by nodes of types (2) and (3). Note that for every node \(C\) in any rooted tree there is a unique critical node \(\text{CCD}(C)\) that is \(m\)’s closest critical descendant, in the sense that \(\text{CCD}(C)\) is a critical node that is a descendant of \(C\) and if \(D\) is any critical node that is a descendant of \(C\) then \(D\) is a descendant of \(\text{CCD}(C)\). Evidently, if \(C\) is itself a critical node then \(\text{CCD}(C) = C\).

A way of providing some evidence that this is not the case is by using Theorem 3 to show that no digital picture \((V', \pi', f')\) which is isomorphic to \((V'', \pi'', f'')\) can be connectedly embedded in the digital picture \((V, \pi, f)\). If a digital picture \((V', \pi', f')\) is connectedly embedded in the digital picture \((V, \pi, f)\), then Theorem 3 implies that there exists a tree embedding of \(T_{(V', \pi', f')}\) into \(T_{(V, \pi, f)}\). It follows that if such a \((V', \pi', f')\) is isomorphic to \((V'', \pi'', f'')\), then there exists a tree embedding of \(T_{(V'', \pi'', f'')}\) into \(T_{(V, \pi, f)}\). However, this cannot happen for the two trees represented
in Figures 3.11(d) and 3.11(a). An easy way to see this is by appealing to the following general result that we state without proof:

**Fact 1.** Let \( \varphi \) be a tree embedding of a rooted tree \( T' = (N', E') \) into a rooted tree \( T = (N, E) \). Let \( M' \) (respectively \( M \)) comprise all the critical nodes of \( N' \) (respectively \( N \)). Define \( \psi : M' \to M \) by \( \psi(m') = \text{CCD}(\varphi(m')) \), for all \( m' \in M' \). It will then be the case, for all \( m'_1 \) and \( m'_2 \) in \( M' \), that \( m'_2 \) is a descendant of \( m'_1 \) in \( T' \) if, and only if, \( \psi(m'_2) \) is a descendant of \( \psi(m'_1) \) in \( T \).

Returning to the tree \( T(V_0, p_0, f_0) \) represented in Figure 3.11(d) and the tree \( T(V, p, f) \) represented in Figure 3.11(a), we observe by Fact 1 that, if there were a tree embedding from \( T(V_0, p_0, f_0) \) into \( T(V, p, f) \), then there would also be a mapping \( \psi \) of critical nodes of \( T(V_0, p_0, f_0) \) (leaves and nodes represented by horizontal lines in Figure 3.11(d)) into the critical nodes of \( T(V, p, f) \) (leaves and nodes represented by horizontal lines in Figure 3.11(a)) that would preserve the descendant relation in the manner specified in Fact 1. However, it is easy to demonstrate that no such mapping \( \psi \) exists.

This shows that no digital picture \( (V', \pi', f') \) which is isomorphic to the digital picture \( (V'', \pi'', f'') \) of Figure 3.10(d) can be connectedly embedded in the digital picture \( (V, \pi, f) \) of Figure 3.10(a). By an analogous argument one can also show that no digital picture \( (V', \pi', f') \) which is isomorphic to the digital picture \( (V'', \pi'', f'') \) of Figure 3.10(b) can be connectedly embedded in the digital picture \( (V, \pi, f) \) of Figure 3.10(c).

The preceding paragraphs have explained how a scientist may be able to determine, by visual inspection of displays of the trees \( T(V'', \pi'', f'') \) and \( T(V, \pi, f) \), that no digital picture which is isomorphic to a picture \( (V'', \pi'', f'') \) can be connectedly embedded in the digital picture \( (V, \pi, f) \).

This cannot, by itself, be regarded as strong evidence that the low-resolution picture \( (V'', \pi'', f'') \) and the high-resolution picture \( (V, \pi, f) \) are pictures of different
things, since many picture embeddings are not connected. But it may also be possible to determine by comparison of the trees $T_{(V',\pi',f')}$ and $T_{(V,\pi,f)}$ that no digital picture which is isomorphic to $(V'',\pi'',f'')$ can be regularly embedded in the digital picture $(V,\pi,f)$, even if the embedding need not be connected. This is because the natural component embedding associated with any such picture embedding would induce a mapping from the nodes of $T_{(V',\pi',f')}$ into the nodes of $T_{(V,\pi,f)}$ that has some easily verified special properties relating to the cardinalities and minimum gray levels of tree nodes\footnote{Since each node of $T_{(V,\pi,f)}$ or $T_{(V',\pi',f')}$ is a set of elements of $V$ or $V''$, the cardinality of a node is just the number of elements of $V$ or $V''$ in that set, and the minimum gray level of a node is the minimum attained by $f$ or $f''$ on that set.} as well as their descendant relationships. It is straightforward to implement algorithms that will determine whether such a mapping of tree nodes exists. If no such mapping exists, then no digital picture which is isomorphic to $(V'',\pi'',f'')$ can be regularly embedded in the digital picture $(V,\pi,f)$.

This illustrates how tree representations can be used to investigate questions concerning whether two images at different resolutions may be representations of the same biological object.

### 3.2 Component Trees of Multidimensional Images

Component trees have been used for numerous applications, including image filtering and segmentation [68], image registration [69], data visualization [70] and pattern recognition [71]. Specifically to the biological field, the component tree has been proposed as a potential tool to explore medical and biological databases [64] and to aid in understanding/exploring the structural information contained in cryo-EM density maps [21, 20]. In this work, component trees will be used for visual exploration of biological data.

The density maps used in the study of a macromolecular complex can be seen
as digital pictures in which the set $V$ contains all the voxels, $\pi$ represents face-adjacency between voxels and $f$ assigns real number values to elements of $V$. One of the challenges in exploring cryo-EM density maps is the noise generated during the processes of specimen preparation and imaging. These processes may change the values assigned to the map and/or modify the structural information within it. In addition, the non-linear stretching of the intensity range in the map and the arbitrary position and orientation of the macromolecule in the map make density map analysis a complex task.

We now head toward showing how the sensitivity of component trees to image noise and structural deformation can be ameliorated by a suitably selected simplification method, which has further uses due to it simplifying the structural information in the density maps. The robust simplification methodology presented in [20] removes nodes from the tree, while preserving the essential structural information of the density map. This simplification is robust in the sense that the method produces essentially the same simplified trees even when the image is slightly perturbed.

The next subsections present a mathematical definition for the foreground component tree structure, a robust tree simplification and an example of using foreground component trees to analyze the structural information of a virus. Even though all the results presented here use foreground component trees, similar results could be found using "background" component trees or simultaneously using foreground and background component trees. A methodology to merge these two kinds of component trees was recently proposed in [72].

### 3.2.1 Foreground Component Tree Structure

Let $T$ be any rooted tree. We write $\text{Nodes}(T)$ to denote the (finite) set of all nodes of $T$, write $\text{root}(T)$ to denote the root of $T$, and write $\text{Leaves}(T)$ to denote the set
Figure 3.12: A Rooted Tree in which the Critical Nodes have been Circled.

In this tree $T$, $\text{Leaves}(T) = \{v_{10}, v_{12}, v_{13}, v_{14}, v_{17}, v_{18}, v_{19}, v_{21}, v_{22}, v_{23}\}$, $\text{root}(T) = v_0$, $\text{Children}_T(v_{11}) = \{v_{17}, v_{18}\}$ and $\text{parent}_T(v_{11}) = v_9$.

of all leaves of $T$. For any $C \in \text{Nodes}(T)$, we write $\text{Children}_T(C)$ to denote the set of all the children of $C$ in $T$ and, if $C \neq \text{root}(T)$, then we write $\text{parent}_T(C)$ to denote the parent of $C$ in $T$. Figure 3.12 illustrates the notation introduced above and the aforementioned critical nodes definition.

Let $(V, \pi)$ be an arbitrary, but fixed, digital space. A foreground component tree structure or FCTS is a pair $(T, \ell)$ for which there exists a collection $\mathcal{C}$ of nonempty $\pi$-connected subsets of $V$ such that the following four conditions hold:

1. $\bigcup \mathcal{C} \in \mathcal{C}$.

2. For all $C, D \in \mathcal{C}$, if $C \not\supset D$ and $C \not\supset D$, then the sets $C$ and $D$ are disjoint and are not $\pi$-adjacent.

3. $\ell$ is a real-valued function on $\mathcal{C}$ such that, for all $C, D \in \mathcal{C}$, $\ell(C) < \ell(D)$ whenever $C \supset D$. (For each $C \in \mathcal{C}$ we call $\ell(C)$ the level of $C$.)

4. $T$ is the rooted tree such that $\text{Nodes}(T) = \mathcal{C}$ and, for all $C, D \in \mathcal{C}$, $C \prec_T D$ if, and only if, $C \supset D$. 
Condition 1 is equivalent to the condition that $C$ has an element which is a super-set of every element of $C$. Moreover, since every element of $C$ is required to be a nonempty $\pi$-connected set, condition 1 implies that $\bigcup C$ is a $\pi$-connected set. Since $\bigcup C$ is finite, $C$ can only be a finite collection. If $C$ is any collection of nonempty finite $\pi$-connected sets that satisfies conditions 1 and 2, and $\ell$ any function that satisfies condition 3, then there will exist a unique rooted tree $T$ that satisfies condition 4 (so that $(T, \ell)$ is an FCTS); the root of this tree will be $\bigcup C$.

**Example 1.** Let $(V, \pi)$ be the digital space where $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and $\pi$ is an adjacency relation on $V$ such that $(n_1, n_2) \in \pi$ if, and only if, $|n_1 - n_2| = 1$. Let $C$ be the following collection of six sets: $\{\{1, 2, 3, 4, 5, 6, 7, 8\}\}, \{1, 2, 3, 4, 5\}, \{1, 2\}, \{4, 5\}, \{7, 8\}, \{8\}\}$. Then it is readily confirmed that $C$ satisfies conditions 1 and 2. Now let $\ell : C \rightarrow \mathbb{R}$ be defined by $\ell(\{1, 2, 3, 4, 5, 6, 7, 8\}) = 12$, $\ell(\{1, 2, 3, 4, 5\}) = 13$, $\ell(\{7, 8\}) = 16$, and $\ell(\{1, 2\}) = \ell(\{4, 5\}) = \ell(\{8\}) = 18$. Then it is readily confirmed that $\ell$ satisfies condition 3. Thus there is an FCTS $(T, \ell)$ for which $\text{Nodes}(T) = C$. The tree $T$ of this FCTS is shown in Figure 3.13.

Note that the component trees in Subsection 3.1.3 can be understood as the rooted trees $T$ for foreground component tree structures $(T, \ell)$. The real-value function $\ell$ is especially relevant in the design of a mutually inverse bijection between the set of all digital pictures and the set of all FCTSs. In what follows, we introduce the notation used to define these maps.
If \( \mathcal{F} \) is an FCTS \((T, \ell)\), then we may use \( \mathcal{F} \) to denote also the rooted tree \( T \) in our terminology and notation. As examples of this, nodes and edges of \( T \) may be referred to as nodes and edges of \( \mathcal{F} \), the notations \( \text{Nodes}(\mathcal{F}) \), \( \text{root}(\mathcal{F}) \), and \( \text{Leaves}(\mathcal{F}) \) will have the same meanings as \( \text{Nodes}(T) \), \( \text{root}(T) \), and \( \text{Leaves}(T) \), and \( \text{parent}_\mathcal{F}(C) \) will have the same meaning as \( \text{parent}_T(C) \) for any \( C \in \text{Nodes}(T) \setminus \text{root}(T) \).

With each digital picture \((V, \pi, f)\) over our (arbitrary, but fixed) digital space \((V, \pi)\), we define the foreground component tree structure \( \text{FCTS}_{(V, \pi, f)} \) by
\[
\text{FCTS}_{(V, \pi, f)} = (T_{(V, \pi, f)}, \ell_{(V, \pi, f)}),
\]
where:

(i) \( T_{(V, \pi, f)} \) is the component tree for the digital picture \((V, \pi, f)\).

(ii) For all \( c \in V \), we have that \( \ell_{(V, \pi, f)}(C_{(V, \pi, f)}(c)) = f(c) \).

Note that \( \ell_{(V, \pi, f)} \) is well defined by this condition, because \( f(c) = f(d) \) whenever \( C_{(V, \pi, f)}(c) = C_{(V, \pi, f)}(d) \). It is readily confirmed that an FCTS with these two properties exists, because \( \mathcal{C} = \{C_{(V, \pi, f)}(c) \mid c \in V\} = C_{(V, \pi, f)}(c) \) satisfies conditions 1 and 2 in the definition of an FCTS; the root of the tree of this FCTS is \( \bigcup \mathcal{C} = V \). It follows from (ii) that for each \( C \in \text{Leaves}(T_{(V, \pi, f)}) \) the level of \( C \) in \( \text{FCTS}_{(V, \pi, f)} \) is just the value of \( f(c) \) in \((V, \pi, f)\) for each element \( c \in C \), and that for each \( C \in \text{Nodes}(T_{(V, \pi, f)}) \) the level of \( C \) is equal to the \( \text{min} \{f(c) \mid c \in C\} \). We call \( \text{FCTS}_{(V, \pi, f)} \) the FCTS of the digital picture \((V, \pi, f)\).

Conversely, we associate with each FCTS \( \mathcal{F} = (T, \ell) \) the digital picture \((V, \pi, f)_{\mathcal{F}}\) that we now define. For each element \( c \in \text{root}(T) \), conditions 2 and 4 in the definition of an FCTS imply that, among the elements of \( \text{Nodes}(T) \) that contain \( c \), there must be a smallest (i.e., a node that is a descendant in \( T \) of every node that contains \( c \)) that element will be denoted by \( \text{node}_T(c) \). We define \( (V, \pi, f)_{\mathcal{F}} = (V, \pi, f)_{(T, \ell)} \) to be the digital picture whose domain is \( \text{root}(T) \), and which satisfies \( f(c) = \ell(\text{node}_T(c)) \) for all \( c \in \text{root}(T) \). We also call \((V, \pi, f)_{\mathcal{F}}\) the
Figure 3.14: The FCTS of a Digital Picture \((V, \pi, f)\).

A reproduction of the digital picture \((V, \pi, f)\) and the component tree \(T = T_{(V, \pi, f)}\) presented in 3.5(a). Writing \((T, \ell)\) for this FCTS, each node of the tree \(T\) is a \(\pi\)-connected set of pixels whose elements are indicated in the figure by the horizontal bar which runs through that node. For example, the root node \(v_0\) of \(T\) consists of all 37 pixels in the domain, the node \(v_1\) consists of all pixels in the domain except the leftmost, and the leaf node \(v_{17}\) consists of just the third and the fourth pixels from the left. For each node \(v\), the value of \(\ell(v)\) can be read from the vertical bar on the left. For example, \(\ell(v_2) = \ell(v_3) = 3\) and \(\ell(v_4) = \ell(v_5) = 6\).

digital picture of the FCTS \(\mathcal{F}\).

Readily, \((V, \pi, f)_{\text{FCTS}_{(V, \pi, f)}} = (V, \pi, f)\) for any digital picture \((V, \pi, f)\) and also \(\text{FCTS}_{(V, \pi, f)} = \mathcal{F}\) for every FCTS \(\mathcal{F}\). Thus the maps \((V, \pi, f) \mapsto \text{FCTS}_{(V, \pi, f)}\) and \(\mathcal{F} \mapsto (V, \pi, f)\) are mutually inverse bijections between the set of all digital pictures and the set of all FCTSs. Consequently, a figure (such as Figure 3.5(a) and repeated in Figure 3.14) that shows an image \((V, \pi, f)\) and its associated FCTS \(\text{FCTS}_{(V, \pi, f)}\) can also be construed as showing the FCTS \(\mathcal{F} = \text{FCTS}_{(V, \pi, f)}\) and its associated image \((V, \pi, f)\).

For the macromolecular docking problem, the mutually inverse bijection property can be useful. Suppose that we are interested in investigating the relationship between nodes from two different FCTS. Since we can obtain the connected component for each node in an FCTS, this investigation can provide information regarding
spell spatial coordinates, component sizes and component volumes.

3.2.2 A Robust \((\lambda,k)\)-Simplification of an FCTS

Unsimplified foreground component trees are too sensitive to error in the image to be good descriptors. Several tree simplification methodologies have been proposed over the years [54, 73, 74, 75]. The core idea of these simplification methodologies are to suppress features that are likely due to noise, artifacts or any other features that are not desired in the tree.

What makes the simplification methodology presented here different from the above-mentioned simplifications is its provable “robustness” in the presence of noise. Suppose we have two digital pictures, for example \((V,\pi,f)\) and \((V,\pi,f')\), such that \((V,\pi,f')\) is a perturbed noise version of \((V,\pi,f)\). It is proved below that under some circumstances, the simplification of \(\text{FCTS}_{(V,\pi,f)}\) is isomorphic to the simplification of \(\text{FCTS}_{(V,\pi,f')}\). The rest of this section presents a careful explanation of the proposed simplification methodology and of the theorem that states its robustness.

Let \(T\) be any rooted tree and \(\mathcal{F} = (T,\ell)\) be an FCTS. Then the set of all critical nodes of \(T\) will be denoted by \(\text{Crit}(T)\) or \(\text{Crit}(\mathcal{F})\). The node in \(\text{Crit}(T)\) that is an ancestor in \(T\) of every node in \(\text{Crit}(T)\) will be called the lowest critical node or LCN of \(T\) or \(\mathcal{F}\), and denoted by \(\text{LCN}(T)\) or \(\text{LCN}(\mathcal{F})\).

For any subset \(S\) of \(\text{Nodes}(T)\) that does not contain every ancestor of \(\text{LCN}(T)\), there is an FCTS \((T',\ell')\) such that \(\text{Nodes}(T') = \text{Nodes}(T) \setminus S\) and \(\ell'\) is the restriction of \(\ell\) to \(\text{Nodes}(T')\). The rooted tree \(T'\) will be denoted by \(T - S\).

We write \(\mathcal{F}' \subseteq \mathcal{F}\) to mean that \(\mathcal{F}' = \mathcal{F} - V\) for some \(V \subseteq \text{Nodes}(T) \setminus \{\text{root}(T)\}\).

Thus \(\mathcal{F}' \subseteq \mathcal{F}\) implies that \(\text{root}(\mathcal{F}') = \text{root}(\mathcal{F})\) and that \(\text{Nodes}(\mathcal{F}') \subseteq \text{Nodes}(\mathcal{F})\).

Let \(T'\text{crit}\) be the rooted tree whose set of nodes is \(\text{Crit}(T) \cup \{\text{root}(T)\}\) and in which a node \(C\) is an ancestor of a node \(D\) if, and only if, \(C\) is an ancestor of
Figure 3.15: Critical Nodes of FCTS

The thick black edges are the edges of the FCTS $\mathcal{F}^{\text{crit}}$, where $\mathcal{F}$ is the FCTS that is shown in Figure 3.17 below. Nodes and edges of $\mathcal{F}$ are shown in gray, but may be hidden by nodes and edges of $\mathcal{F}^{\text{crit}}$—for example, the edges of $\mathcal{F}$ that join $v_4$ to $v_8$ and $v_8$ to $v_{12}$ in Figure 3.17 are not visible in this figure because they are hidden by the edge of $\mathcal{F}^{\text{crit}}$ that joins $v_4$ to $v_{12}$. (The digital picture $(V, \pi, f)_{\mathcal{F}^{\text{crit}}}$ of $\mathcal{F}^{\text{crit}}$ is shown at the top.)

$D$ in $T$. Thus $\text{root}(T^{\text{crit}}) = \text{root}(T)$, $\text{LCN}(T^{\text{crit}}) = \text{LCN}(T)$, and $\text{Crit}(T^{\text{crit}}) = \text{Crit}(T)$. If $\text{LCN}(T) \neq \text{root}(T)$, then $\text{LCN}(T^{\text{crit}}) = \text{LCN}(T)$ is the unique child of $\text{root}(T^{\text{crit}}) = \text{root}(T)$ in $T^{\text{crit}}$. The FCTS $(T^{\text{crit}}, \ell^{\text{crit}})$, where $\ell^{\text{crit}}$ is the restriction of $\ell$ to $\text{Nodes}(T^{\text{crit}})$, will be denoted by $\mathcal{F}^{\text{crit}}$. Note that $T^{\text{crit}} \subseteq T$. This concept is illustrated in Figures 3.15 and 3.17.

Using this terminology, the method of simplifying $\text{FCTS}_{(V, \pi, f)}$ can be stated as follows:

Let $\mathcal{F}_0 = (T_0, \ell_0)$ be any FCTS. Then, for every positive real value $\lambda$ and every nonnegative integer $k < |\text{root}(T_0)|$, we define the $(\lambda, k)$-simplification of $\mathcal{F}_0$ to be the component tree $\mathcal{F}_3$ that can be obtained from $\mathcal{F}_0$ in three steps, as follows:

Step 1: Prune $\mathcal{F}_0$ by removing nodes of size $\leq k$, to produce $\mathcal{F}_1 \subseteq \mathcal{F}_0$.

Step 2: Prune $\mathcal{F}_1$ by removing branches of length $\leq \lambda$, to produce $\mathcal{F}_2 \subseteq \mathcal{F}_1$. 

A similar method can be used with the FCTS $\mathcal{F}^{\text{crit}}$. Let $\mathcal{F}^{\text{crit}} = (T^{\text{crit}}, \ell^{\text{crit}})$ be any FCTS. Then, for every positive real value $\lambda$ and every nonnegative integer $k < |\text{root}(T^{\text{crit}})|$, we define the $(\lambda, k)$-simplification of $\mathcal{F}^{\text{crit}}$ to be the component tree $(T^{\text{crit}}, \ell^{\text{crit}}, k)$ that can be obtained from $\mathcal{F}^{\text{crit}}$ in three steps, as follows:

Step 3: Prune $\mathcal{F}^{\text{crit}}$ by removing nodes of size $\leq k$, to produce $\mathcal{F}^{\text{crit}'} \subseteq \mathcal{F}^{\text{crit}}$.

Step 4: Prune $\mathcal{F}^{\text{crit}'}$ by removing branches of length $\leq \lambda$, to produce $\mathcal{F}^{\text{crit}''} \subseteq \mathcal{F}^{\text{crit}'}$. 

Note that $\mathcal{F}^{\text{crit}''} \subseteq \mathcal{F}^{\text{crit}}$. 

This concept is illustrated in Figures 3.15 and 3.17.
Step 3: Eliminate internal edges of length $\leq \lambda$ from $\mathcal{F}^{\text{crit}}_2$, to produce the final FCTS $\mathcal{F}_3 \subseteq \mathcal{F}^{\text{crit}}_2$.

With the possible exception of the root, every non-leaf node of the final FCTS is a critical node both of $\mathcal{F}_3$ and of the original FCTS $\mathcal{F}_0$. We now describe more precisely the steps from a $(\lambda, k)$-simplification.

**Step 1 - Removing Small Components:**

Step 1 is one of the filtering methods proposed in Section VI of [63]. It is defined as follows: The result of pruning the FCTS $\mathcal{F}_0 = (T_0, \ell_0)$ by removing nodes of size $\leq k$ is just the FCTS

$$\mathcal{F}_0 - \{C \in \text{Nodes}(\mathcal{F}_0) \mid |C| \leq k\},$$

where, as usual, $|C|$ denotes the cardinality of the set $C$; i.e., the number of spels in $C$. Note that the result is just $\mathcal{F}_0$ itself if $k = 0$. Figure 3.16 shows an FCTS that has been obtained by pruning the FCTS of Figure 3.5(a) and 3.14 in this way.

Precise definitions of steps 2 and 3 of $(\lambda, k)$-simplification will be given in Subsections 3.2.4 and 3.2.5 below.

### 3.2.3 Essential Isomorphism and Level-Preserving Theorem

While the simplification method presented in the last subsection is somewhat similar to the method of [64], it has the robustness properties that are stated in Theorem 4 and Corollary 3 below (which the method of [64] does not have). We now introduce terminology and notation that will be used to state these two results.

We say that two FCTSs $\mathcal{F}_a = (T_a, \ell_a)$ and $\mathcal{F}_b = (T_b, \ell_b)$ are *essentially isomorphic* if the subtree of $T_a^{\text{crit}}$ that is rooted at $\text{LCN}(T_a)$ is isomorphic to the subtree of $T_b^{\text{crit}}$ that is rooted at $\text{LCN}(T_b)$. Thus $\mathcal{F}_a$ and $\mathcal{F}_b$ are essentially isomorphic if,
Figure 3.16: The Effect of Pruning an FCTS by Removing Nodes of Size $\leq k$.

The effect of pruning the FCTS of Figure 3.14 by removing nodes of size $\leq k$ is shown for the case $k = 1$; the black edges are the edges of the resulting FCTS. Just two nodes ($v_{10}$ and $v_{23}$) are removed from the tree of Figure 3.14, as these are the only nodes of that tree that consist of no more than $k$ pixels (i.e., no more than 1 pixel, since $k = 1$). The image of the resulting FCTS is shown at the top. Note that the graylevel of the second pixel from the right has changed from 18 in Figure 3.14 to 16 here; this reflects the removal of $v_{23}$ from the tree. Similarly, the graylevel of the 17th pixel from the left has changed from 10 to 9; this reflects the removal of $v_{10}$. The graylevels of the other 35 pixels are the same as in Figure 3.14.

and only if, there exists a mapping $\theta : \text{Crit}(T_a) \rightarrow \text{Crit}(T_b)$ such that $\theta(\text{Crit}(T_a)) = \text{Crit}(T_b)$ and, for all $C, D \in \text{Crit}(T_a), C \preceq T_a D$ if, and only if, $\theta(C) \preceq T_b \theta(D)$. (The latter property implies that $\theta$ is 1-to-1.) Any such $\theta$ will be called an essential isomorphism of $\mathcal{F}_a$ to $\mathcal{F}_b$.

Note that if the rooted trees $T_a^{\text{crit}}$ and $T_b^{\text{crit}}$ are isomorphic, then $\mathcal{F}_a = (T_a, \ell_a)$ and $\mathcal{F}_b = (T_b, \ell_b)$ are certainly essentially isomorphic. The converse is almost but not quite true. The only way in which $\mathcal{F}_a = (T_a, \ell_a)$ and $\mathcal{F}_b = (T_b, \ell_b)$ could be essentially isomorphic without $T_a^{\text{crit}}$ and $T_b^{\text{crit}}$ being isomorphic is if the root is the same as the LCN in one of the trees but not in the other, and when we remove the root from the latter tree (so its LCN becomes its root) it becomes isomorphic to the
Figure 3.17: Example of $K_{(V, \pi, f)}$ and $\Lambda_{(V, \pi, f)}$.

If $(V, \pi, f)$ is the digital picture at the top ($\pi$ is the same adjacency relation as in Figures 3.14 and 3.16), then $\Lambda_{(V, \pi, f)} = 5$ and $K_{(V, \pi, f)} = 2$. $\Lambda_{(V, \pi, f)} = 5$ because, writing $(T, \ell)$ for the FCTS of $(V, \pi, f)$ (which is shown in this figure), $T$ has critical nodes $v_i$ and $v_j$ such that $v_i \succ_T v_j$ and $\ell(v_i) - \ell(v_j) = 5$ (e.g., $(v_i, v_j) = (v_4, v_1)$), but $T$ has no critical nodes $v_i$ and $v_j$ such that $v_i \succ_T v_j$ and $\ell(v_i) - \ell(v_j) < 5$. $K_{(V, \pi, f)} = 2$ because $T$ has a node (e.g., $v_9$) that consists of just 2 pixels, but no node of $T$ consists of fewer than 2 pixels.

For any $\delta \geq 0$, if an essential isomorphism $\theta$ of $\mathcal{F}_a$ to $\mathcal{F}_b$ satisfies the condition $|\ell_b(\theta(C)) - \ell_a(C)| \leq \delta$ for all $C \in \text{Crit}(\mathcal{F}_a)$, then we say that $\theta$ is level-preserving to within $\delta$. Evidently, the inverse of any essential isomorphism of $\mathcal{F}_a$ to $\mathcal{F}_b$ that is level-preserving to within $\delta$ will be an essential isomorphism of $\mathcal{F}_b$ to $\mathcal{F}_a$ that is level-preserving to within $\delta$.

If an essential isomorphism $\theta$ of $\mathcal{F}_a$ to $\mathcal{F}_b$ is level-preserving to within 0 (i.e., if $\ell_b(\theta(C)) = \ell_a(C)$ for all $C \in \text{Crit}(\mathcal{F}_a)$), then we say that $\theta$ is level-preserving.

**Example 2.** The FCTS shown in Figure 3.17 is essentially isomorphic to the FCTS shown by the thick black edges in Figure 3.19. Indeed, if $(T, \ell)$ is the FCTS shown in Figure 3.17, and $(T_*, \ell_*)$ is the FCTS shown by the thick black edges in Figure 3.19, then $(T, \ell)_{\text{crit}}$ is the FCTS shown in Figure 3.15, and $(T_*, \ell_*)_{\text{crit}} = (T_*, \ell_*)$. 
It is evident from a quick glance at Figures 3.15 and 3.19 that $T^{\text{crit}}$ is isomorphic to $T^{\text{crit}}_* = T_*$, so that $(T, \ell)$ is essentially isomorphic to $(T_*, \ell_*)$, as we claimed. It is readily confirmed that the mapping $\theta : \text{Crit}(T) \to \text{Crit}(T_*)$ which respectively maps

\[ V_{1, 4, 5, 9, 10, 11, 12, 14, 15, 16, 17} \quad \text{in Figure 3.17 (or Figure 3.15)} \]

to

\[ V_{1, 4, 5, 13, 14, 15, 17, 19, 20, 21, 22} \quad \text{in Figure 3.19} \]

is an essential isomorphism of $(T, \ell)$ to $(T_*, \ell_*)$. The essential isomorphism $\theta$ is not level-preserving, since $|\ell_*(\theta(C)) - \ell(C)| = 1$ when $C = v_{12}$ and when $C = v_{15}$; indeed, $\ell(v_{12}) = 13$, but $\ell_*(\theta(v_{12})) = 14$, and $\ell(v_{15}) = 17$ but $\ell_*(\theta(v_{15})) = 16$. But it is readily confirmed that $\ell_*(\theta(C)) = \ell(C)$ for all $C \in \text{Crit}(T) \setminus \{v_{12}, v_{15}\}$, and so $\theta$ is level-preserving to within 1.

Let $(V, \pi, f)$ be a digital picture and let $(T_{(V,\pi,f)},\ell_{(V,\pi,f)}) = \text{FCTS}_{(V,\pi,f)}$. Then we define:

\[
K_{(V,\pi,f)} = \min_{c \in V} |C_{(V,\pi,f)}(c)| = \min_{C \in \text{Leaves}(T_{(V,\pi,f)})} |C|
\]

\[
\Lambda_{(V,\pi,f)} = \min \{ \ell_{(V,\pi,f)}(C) - \ell_{(V,\pi,f)}(D) \mid C, D \in \text{Crit}(T_{(V,\pi,f)}) \text{ and } C \succ T_{(V,\pi,f)} D \}
\]

These concepts are illustrated in Figure 3.17.

If $(V, \pi, f)$ and $(V, \pi, f')$ are two digital pictures, then the value $\max_{c \in V} |f'(c) - f(c)|$ will be denoted by $\|f' - f\|_\infty$.

Using this notation, we now state our principal robustness result regarding $(\lambda, k)$-simplification (a result which we will generalize in Corollary 3):

**Theorem 4.** Let $(V, \pi, f)$ be any digital picture, $k$ any integer such that $0 \leq k < K_{(V,\pi,f)}$, and $\lambda$ any value such that $0 < \lambda < \Lambda_{(V,\pi,f)}/2$. Let $(V, \pi, f')$ be a digital picture such that $\|f' - f\|_\infty \leq \lambda/2$. Then there is an essential isomorphism of...
the \((\lambda, k)\)-simplification of \(\text{FCTS}_{(V, \pi, f')}\) to \(\text{FCTS}_{(V, \pi, f)}\) that is level-preserving to within \(\lambda/2\).

A proof of this theorem is given in Appendix B. In Theorem 4, and in Corollary 3 below, we may think of the digital picture \((V, \pi, f)\) as an ideal or perfect digital picture of some object (such as a macromolecule) at a certain level of detail/resolution, and think of the image \((V, \pi, f')\) as an imperfect noisy approximation to the ideal digital picture \((V, \pi, f)\) (such as an EM map of the same object).

We may suppose that the ideal digital picture \((V, \pi, f)\) is not available to us (and we do not know the exact structure of \(\text{FCTS}_{(V, \pi, f)}\)), but the imperfect digital picture \((V, \pi, f')\) is available and we can therefore construct \(\text{FCTS}_{(V, \pi, f')}\). Theorem 4 and Corollary 3 assure us that, if \((V, \pi, f')\) is “sufficiently similar” to \((V, \pi, f)\), then there will be values of \(\lambda\) and \(k\) for which the \((\lambda, k)\)-simplification of \(\text{FCTS}_{(V, \pi, f')}\) is essentially isomorphic to \(\text{FCTS}_{(V, \pi, f)}\).

For this purpose, it follows from Theorem 4 that the imperfect noisy approximation \((V, \pi, f')\) will be “sufficiently similar” to the ideal digital picture \((V, \pi, f)\) if there is no spel in \(V\) at which the value of \((V, \pi, f')\) differs from the value of \((V, \pi, f)\) by \(\Lambda_{(V, \pi, f)}/4\) or more. Additionally, it will follow from Corollary 3 that \((V, \pi, f')\) might be sufficiently similar to \((V, \pi, f)\) even if this condition is violated at a small number of spels whose values in \((V, \pi, f)\) and \((V, \pi, f')\) may differ by arbitrarily large amounts.

**Example 3.** To illustrate Theorem 4, let \((V, \pi, f)\) be the digital picture that is shown in Figure 3.17 and let \((V, \pi, f')\) be the digital picture image that is shown in Figure 3.14. Then \(\|f' - f\|_{\infty} = 1\), because there exists a pixel \(p\) (e.g., any of the three rightmost pixels in the domain) for which \(|f'(p) - f(p)| = 1\), but there is no pixel \(p\) for which \(|f'(p) - f(p)| > 1\). Now let \(\lambda = 2\) and \(k = 1\). As we observe in the caption of Figure 3.17, \(\Lambda_{(V, \pi, f)} = 5\) and \(K_{(V, \pi, f)} = 2\), so the conditions \(\lambda < \Lambda_{(V, \pi, f)}/2\), \(k < K_{(V, \pi, f)}\), and \(\|f' - f\|_{\infty} \leq \lambda/2\) that appear in Theorem 4 are satisfied. Thus
the theorem says that there is an essential isomorphism of the \((\lambda, k)\)-simplification of \(\text{FCTS}_{(V, \pi, f')}\) to \(\text{FCTS}_{(V, \pi, f)}\) that is level-preserving to within \(\lambda/2 = 1\). In fact the inverse of the mapping \(\theta\) defined in Example 2 above is just such an essential isomorphism! That is, (as we will see in Section 3.2.5) the FCTS shown by the thick black edges in Figure 3.19 is exactly the \((\lambda, k)\)-simplification of \(\text{FCTS}_{(V, \pi, f')}\).

From Theorem 4, it is easy to deduce Corollary 3 below. Theorem 4 is essentially the case of Corollary 3 in which \(k^* = 0\) and \((V, \pi, f^*) = (V, \pi, f)\).

As mentioned above, one can think of \((V, \pi, f)\) in Theorem 4 and Corollary 3 as a perfect or ideal digital picture, and think of \((V, \pi, f')\) as an imperfect approximation to \((V, \pi, f)\). Theorem 4 is applicable only if the graylevel of every spel in \((V, \pi, f')\) is close to (specifically, within less than \(\Lambda_{(V, \pi, f)}/4\) of) that spel’s graylevel in \((V, \pi, f)\). Corollary 3 is more general, and it may be applicable even if there are exceptional spels at which \((V, \pi, f')\) is graylevel is much lower or higher than \((V, \pi, f)\)’s graylevel.

**Corollary 3.** Let \((V, \pi, f)\) and \((V, \pi, f')\) be digital pictures. For any nonnegative integer \(k < |V|\), let \((V, \pi, f'_k)\) denote the digital picture of the FCTS that results from pruning \(\text{FCTS}_{(V, \pi, f')}\) by removing nodes of size \(\leq k\). Suppose there is a digital picture \((V, \pi, f^*)\) such that there exists a level-preserving essential isomorphism of \(\text{FCTS}_{(V, \pi, f^*)}\) to \(\text{FCTS}_{(V, \pi, f)}\) and there exists a nonnegative integer \(k^* < K_{(V, \pi, f^*)}\) for which the digital picture \((V, \pi, f^*)\) satisfies \(\|f'_k - f^*\|_\infty < \Lambda_{(V, \pi, f)}/4\). Then, for any positive \(\lambda\) and integer \(k\) such that \(2\|f - f^*\|_\infty \leq \lambda < \Lambda_{(V, \pi, f)}/2\) and \(k^* < k < K_{(V, \pi, f^*)}\), there is an essential isomorphism of the \((\lambda, k^*)\)-simplification of \(\text{FCTS}_{(V, \pi, f')}\) to \(\text{FCTS}_{(V, \pi, f)}\) that is level-preserving to within \(\lambda/2\).

**Proof.** (Assuming Theorem 4) Let \(k\) be an integer such that \(k^* \leq k < K_{(V, \pi, f^*)}\) and \(\lambda\) a positive value such that \(2\|f'_k - f\|_\infty \leq \lambda < \Lambda_{(V, \pi, f)}/2\).

Now \(\text{FCTS}_{(V, \pi, f'_k)}\) is the result of applying step 1 of \((\lambda, k^*)\)-simplification to
\textbf{FCTS}_{(V, \pi, f')} \). It follows that the \((\lambda, k)\)-simplification of \textbf{FCTS}_{(V, \pi, f')} is the same as the \((\lambda, k)\)-simplification of \textbf{FCTS}_{(V, \pi, f'_k)} (since applying simplification step 1 twice in succession with parameter \(k^*\) and then \(k\) has the same effect as applying step 1 just once with the parameter \(\max(k^*, k) = k\)). To prove the corollary, we need to show that there is an essential isomorphism of this \textbf{FCTS} (i.e., the \((\lambda, k)\)-simplification of \textbf{FCTS}_{(V, \pi, f'_k)} to \textbf{FCTS}_{(V, \pi, f)} that is level-preserving to within \(\lambda/2\).

We have that \(\Lambda_{(V, \pi, f)} = \Lambda_{(V, \pi, f^*)}\), since there is a level-preserving essential isomorphism of \textbf{FCTS}_{(V, \pi, f^*)} to \textbf{FCTS}_{(V, \pi, f)}. Thus we have that \(\lambda < \Lambda_{(V, \pi, f^*)}/2\). Moreover, \(\|f'_k - f^*\|_{\infty} \leq \lambda/2\) and \(k < K_{(V, \pi, f^*)}\). On applying Theorem 4 to \((V, \pi, f^*)\) and \((V, \pi, f'_k)\), we see that there is an essential isomorphism of the \((\lambda, k)\)-simplification of \textbf{FCTS}_{(V, \pi, f'_k)} to \textbf{FCTS}_{(V, \pi, f^*)} that is level-preserving to within \(\lambda/2\). Composing this essential isomorphism with the level-preserving essential isomorphism of \textbf{FCTS}_{(V, \pi, f^*)} to \textbf{FCTS}_{(V, \pi, f)} gives an essential isomorphism of the \((\lambda, k)\)-simplification of \textbf{FCTS}_{(V, \pi, f'_k)} onto \textbf{FCTS}_{(V, \pi, f)} that is level-preserving to within \(\lambda/2\), as required.

The following example shows how the condition that \((V, \pi, f')\) must satisfy in Corollary 3 is much less restrictive than the condition \(\|f' - f\|_{\infty} \leq \Lambda_{(V, \pi, f)}/4\) that \((V, \pi, f')\) needs to satisfy for Theorem 4 to be applicable.

\textbf{Example 4.} Let \(V\) be a 3D rectangular array of voxels, and let \(\pi\) be the 6-adjacency relation on \(V\). Let \((V, \pi, f)\) be a digital picture such that, for each threshold \(\tau \leq \max_{v \in V} f(v)\), the members of \(\{C_{(V, \pi, f)}(v) \mid f(v) \geq \tau\}\) have fairly compact shapes, are not very small and no two of the sets are very close together. (Here “have fairly compact shapes” and “are not very small” imply that: (i) removing a very few randomly chosen voxels from a set \(C_{(V, \pi, f)}(v)\) is unlikely to split it into two or more pieces and unlikely to completely eliminate that set. The “no two of the sets are very close” condition implies that: (ii) adding a very few randomly chosen voxels
to a set \( C_{(V,\pi,f)}(v) \) is unlikely to connect that set to a different set \( C_{(V,\pi,f')}() \). Now let \( (V,\pi,f') \) be a digital picture on \( V \) that is obtained from \( (V,\pi,f) \) by changing the graylevels of a very small number of randomly chosen voxels by arbitrarily large positive and/or negative amounts. Then \( \|f' - f\|_\infty \leq \Lambda_{(V,\pi,f)}/4 \) will not hold unless every graylevel change is smaller in absolute value than \( \Lambda_{(V,\pi,f)}/4 \). But, regardless of the sizes of the graylevel changes, when \( k^* \) is the cardinality of the largest 6-connected subset of the set \( \{v \in V \mid f'(v) > f(v)\} \) it is likely (because of (i) and (ii)) that there will be a level-preserving essential isomorphism of \( \text{FCTS}_{(V,\pi,f')} \) to \( \text{FCTS}_{(V,\pi,f)} \), in which case the image \( (V,\pi,f') \) will satisfy the condition of Corollary 3 with \( (V,\pi,f^*) = (V,\pi,f'_{k^*}) \).

### 3.2.4 Pruning by Removing Branches of Length \( \leq \lambda \)

Step 2 of \((\lambda,k)\)-simplification is to prune the FCTS that is the result of step 1 by removing branches of length \( \leq \lambda \). This subsection presents a mathematical specification of the output of step 2 (properties P1 – P4 below), a result (Proposition 4) that gives us an easily visualized characterization of the output, followed by a description (in Sub-subsection 3.2.4.3) of how step 2 can be efficiently implemented.

#### 3.2.4.1 Specification of Simplification Step 2

Let \( T \) be any rooted tree and let \( C \in \text{Nodes}(T) \). We write \( C \upharpoonright_T \) to denote the set of all ancestors of \( C \) in \( T \), \( C \uparrow_T \) to denote the set \( C \downarrow_T \setminus \{C\} \) (i.e., the set of all proper ancestors of \( C \) in \( T \)), and \( C \uparrow_T \) to denote the set of all descendants of \( C \) in \( T \), and \( C \downarrow_T \) to denote the set \( C \uparrow_T \setminus \{C\} \) (i.e., the set of all proper descendants of \( C \) in \( T \)).

Now let \( \emptyset \neq S \subseteq \text{Nodes}(T) \). Then we write \( \bigwedge_T S \) to denote the closest common ancestor of \( S \), by which we mean the node \( D \) of \( T \) such that \( D \downarrow_T = \bigcap_{C \in S} C \downarrow_T \), or, equivalently, the element of \( \bigcap_{C \in S} C \downarrow_T \) that is a descendant in \( T \) of every element of that set.
For any FCTS $\mathcal{F}_{in} = (T_{in}, \ell_{in})$, we call a sequence $\text{leaf}[1], \ldots, \text{leaf}[n]$ an $\ell_{in}$-increasing enumeration of $\text{Leaves}(\mathcal{F}_{in})$ if no two elements of $\text{leaf}[1], \ldots, \text{leaf}[n]$ are the same, $\{\text{leaf}[1], \ldots, \text{leaf}[n]\} = \text{Leaves}(\mathcal{F}_{in})$ (so that $n = |\text{Leaves}(\mathcal{F}_{in})|$), and $\ell_{in}[\text{leaf}[1]] \leq \cdots \leq \ell_{in}[\text{leaf}[n]]$. Pruning an FCTS $\mathcal{F}_{in}$ by removing branches of length $\leq \lambda$ is done using such an enumeration of $\text{Leaves}(\mathcal{F}_{in})$.

For any $\lambda > 0$, any FCTS $\mathcal{F}_{in}$, and any $\ell_{in}$-increasing enumeration $\text{leaf}[1], \ldots, \text{leaf}[n]$ of $\text{Leaves}(\mathcal{F}_{in})$, we define the result of pruning $\mathcal{F}_{in}$ by removing branches of length $\leq \lambda$ using the leaf enumeration $\text{leaf}[1], \ldots, \text{leaf}[n]$ to be the FCTS $\mathcal{F}_{out}$ that has the following four properties:

P1: $\mathcal{F}_{out} \subseteq \mathcal{F}_{in}$

P2: $\text{leaf}[n] \in \text{Leaves}(\mathcal{F}_{out})$

P3: For $1 \leq i < n$, $\text{leaf}[i] \in \text{Leaves}(\mathcal{F}_{out})$ if, and only if, there does not exist any $j \in \{i + 1, \ldots, n\}$ for which $\ell_{in}(\text{leaf}[i]) - \ell_{in}(\bigwedge_{\mathcal{F}_{in}} \{\text{leaf}[j], \text{leaf}[i]\}) \leq \lambda$.

P4: $\text{Nodes}(\mathcal{F}_{out}) = \bigcup \{\text{leaf}[i] \downarrow_{\mathcal{F}_{in}} | 1 \leq i \leq n \text{ and } \text{leaf}[i] \in \text{Leaves}(\mathcal{F}_{out})\}$

Given any FCTS $\mathcal{F}_{in}$, any $\lambda > 0$, and any $\ell_{in}$-increasing enumeration $\text{leaf}[1], \ldots, \text{leaf}[n]$ of $\text{Leaves}(\mathcal{F}_{in})$, it is evident that P1–P4 uniquely determine $\mathcal{F}_{out}$. Moreover, even though the result $\mathcal{F}_{out}$ of pruning may depend on the leaf enumeration $\text{leaf}[1], \ldots, \text{leaf}[n]$ that is used, we will have that for any given $\mathcal{F}_{in}$ and $\lambda$, P1–P4 uniquely determine $\mathcal{F}_{out}$ up to a level-preserving essential isomorphism.

Figure 3.18 shows an FCTS that has been obtained by pruning the FCTS of Figure 3.16 in this way.

### 3.2.4.2 An Easily Visualized Characterization of the Output of Simplification

#### Step 2

The main goal of this sub-subsection is to present a result (Proposition 4) that is important for the following reasons:
Figure 3.18: The Effect of Pruning an FCTS by Removing Branches of Length $\lambda$. The effect of pruning an FCTS of Figure 3.16 by removing branches of length $\lambda$ is shown, in the case $\lambda = 2$; the black edges are the edges of the resulting FCTS. Writing $(T_1, \ell_1)$ for the FCTS of Figure 3.16, it is assumed that pruning is done using an $\ell_1$-increasing leaf enumeration in which the leaf $v_{17}$ of $T_1$ occurs later than the leaf $v_{18}$. The leaves $v_8$, $v_{12}$, and $v_{18}$ are the only nodes of $T_1$ that are removed; the leaf $v_8$ is removed because we have that $\ell_1(v_8) = \ell_1(v_9) - \ell_1(v_7) \leq 2 = \lambda$ (and $v_{19}$ occurs later in the $\ell_1$-increasing leaf enumeration than $v_8$ because $\ell_1(v_8) < \ell_1(v_{19})$); $v_{12}$ is removed because $\ell_1(v_{12}) = \ell_1(v_9) - \ell_1(v_{13}) = 2 = \lambda$; $v_{18}$ is removed because $\ell_1(v_{18}) = \ell_1(v_{11}) - \ell_1(v_{18}) = 2 = \lambda$ and we are assuming (as mentioned above) that $v_{17}$ occurs later in the $\ell_1$-increasing leaf enumeration than $v_{18}$. In this example, no non-leaf nodes of $T_1$ are removed, as every non-leaf node of $T_1$ is an ancestor of a leaf of $T_1$ that is not removed.

1. It shows that the output of step 2 is independent of the leaf enumeration which is used for pruning (up to a level-preserving essential isomorphism).

2. It gives an easily visualized characterization of the output. (This will be further explained after Proposition 4.)

3. The linear-time implementation of step 2 that is described in Sub-subsection 3.2.4.3 is based on this result.
For any rooted tree $T$ and any $C \in \text{Nodes}(T)$, we write $T_{[C]}$ to denote the subtree of $T$ that is rooted at $C$.

Now we define some other notation that will be used in Proposition 4. For this purpose, let $\mathcal{F} = (T, \ell)$ be any FCTS and $\lambda$ any positive value. Then we define $\text{depth}_{\mathcal{F}}(C) = \max_{D \in \text{Leaves}(T_{[C]})} \ell(D) - \ell(C)$. Note that $\text{depth}_{\mathcal{F}}(C) = 0$ for all $C \in \text{Leaves}(T)$. We also define:

$$\mathcal{U}_\lambda^\lambda(\mathcal{F}) = \{ C \in \text{Nodes}(T) \mid \text{depth}_{\mathcal{F}}(C) > \lambda \}$$

$$\mathcal{V}_\lambda^\lambda(\mathcal{F}) = \{ C \in \text{Nodes}(T) \mid C \notin \mathcal{U}_\lambda^\lambda(\mathcal{F}) \text{ but } C \uparrow T \subseteq \mathcal{U}_\lambda^\lambda(\mathcal{F}) \}$$

If $\mathcal{U}_\lambda^\lambda(\mathcal{F}) \neq \emptyset$, then $C \in \mathcal{V}_\lambda^\lambda(\mathcal{F})$ if, and only if, $C \in \text{root}(T) \uparrow T$, $\text{depth}_{\mathcal{F}}(C) \leq \lambda$, and $\text{depth}_{\mathcal{F}}(\text{parent}_T(C)) > \lambda$. If $\mathcal{U}_\lambda^\lambda(\mathcal{F}) = \emptyset$, then $\mathcal{V}_\lambda^\lambda(\mathcal{F}) = \{ \text{root}(T) \}$.

For any $C \in \text{Nodes}(T)$, either $C \in \mathcal{U}_\lambda^\lambda(\mathcal{F})$ or $C$ has a unique ancestor in $\mathcal{V}_\lambda^\lambda(\mathcal{F})$ (possibly itself), and $C$ satisfies just one of those conditions. Hence:

$$\text{Nodes}(T) = \mathcal{U}_\lambda^\lambda(\mathcal{F}) \cup \bigcup_{C \in \mathcal{V}_\lambda^\lambda(\mathcal{F})} C \uparrow T$$  \hspace{1cm} (3.13)

If $\mathcal{U}_\lambda^\lambda(\mathcal{F}) \neq \emptyset$ (so that $\text{root}(T)$ lies in $\mathcal{U}_\lambda^\lambda(\mathcal{F})$ and not in $\mathcal{V}_\lambda^\lambda(\mathcal{F})$), then we define:

$$\mathcal{V}_1^\lambda(\mathcal{F}) = \{ C \in \mathcal{V}_\lambda^\lambda(\mathcal{F}) \mid \text{depth}_{\mathcal{F}}(C) + \ell(C) - \ell(\text{parent}_T(C)) > \lambda \}$$

But if $\mathcal{U}_\lambda^\lambda(\mathcal{F}) = \emptyset$, then we define $\mathcal{V}_1^\lambda(\mathcal{F}) = \{ \text{root}(T) \} = \mathcal{V}_\lambda^\lambda(\mathcal{F})$.

Let $\sigma = (\text{leaf}[1], \ldots, \text{leaf}[n])$ be any $\ell$-increasing enumeration of the leaves of the tree $T$, and $C$ any node of $T$. Then we define lastLeaf$^\sigma_T(C, T)$ to be the leaf of $T_{[C]}$ that occurs later in the $\ell$-increasing enumeration $\sigma$ than all other leaves of $T_{[C]}$. (If $T_{[C]}$ has just one leaf, then lastLeaf$^\sigma_T(C, T)$ is that leaf.) Thus we have that $\text{depth}_{\mathcal{F}}(C) = \ell(\text{lastLeaf}_T(C, T)) - \ell(C)$. We define Path$^\sigma_T(C, T) = \{ D \in \text{Nodes}(T) \mid C \preceq_T D \preceq \text{lastLeaf}_T(C, T) \}$. (Note that if $C'$ is any node of $T$ that is nei-
ther an ancestor nor a descendant of \(C\) in \(T\), then \(\text{lastLeaf}_\sigma(C, T) \neq \text{lastLeaf}_\sigma(C', T)\) and \(\text{Path}_\sigma(C, T) \cap \text{Path}_\sigma(C', T) = \emptyset\).

Using the notation previously introduced, the main result of this section, which is proved in Appendix A, is now introduced.

**Proposition 4.** Let \(\mathcal{F}_{\text{in}} = (T_{\text{in}}, \ell_{\text{in}})\) be any FCTS, let \(\lambda > 0\), and let \(\mathcal{F}_{\text{out}} = (T_{\text{out}}, \ell_{\text{out}})\) be the FCTS that results from pruning \(\mathcal{F}_{\text{in}}\) by removing branches of length \(\leq \lambda\) using an \(\ell_{\text{in}}\)-increasing enumeration \(\sigma\) of \(\text{Leaves}(T_{\text{in}})\). Then the nodes of \(\mathcal{F}_{\text{out}}\) consist just of:

(i) The nodes of \(U^\lambda(\mathcal{F}_{\text{in}})\).

(ii) The nodes of \(\text{Path}_\sigma(C, T_{\text{in}})\) for each node \(C\) in \(V^\lambda_1(\mathcal{F}_{\text{in}})\).

Now let \(\mathcal{F}_{\text{in}} = (T_{\text{in}}, \ell_{\text{in}}), \lambda, \sigma,\) and \(\mathcal{F}_{\text{out}} = (T_{\text{out}}, \ell_{\text{out}})\) be as in Proposition 4. Since \(V^\lambda_1(\mathcal{F}_{\text{in}}) \subseteq V^\lambda(\mathcal{F}_{\text{in}})\), and since no node in \(V^\lambda(\mathcal{F}_{\text{in}})\) is an ancestor in \(T_{\text{in}}\) of a node in \(U^\lambda(\mathcal{F}_{\text{in}})\) or of a different node in \(V^\lambda(\mathcal{F}_{\text{in}})\), for all \(C \in V^\lambda_1(\mathcal{F}_{\text{in}})\) we have that \(\text{Path}_\sigma(C, T_{\text{in}}) \cap U^\lambda(\mathcal{F}_{\text{in}}) = \emptyset\), and for all distinct \(C, C' \in V^\lambda_1(\mathcal{F}_{\text{in}})\) we have that \(\text{Path}_\sigma(C, T_{\text{in}}) \cap \text{Path}_\sigma(C', T_{\text{in}}) = \emptyset\).

Thus Proposition 4 gives us an easily visualized characterization of the nodes of the FCTS \(\mathcal{F}_{\text{out}} = (T_{\text{out}}, \ell_{\text{out}})\) that results from pruning \(\mathcal{F}_{\text{in}}\) by removing branches of length \(\leq \lambda\) using the leaf enumeration \(\sigma\) (and hence an easily visualized characterization of \(\mathcal{F}_{\text{out}}\) itself, since \(\mathcal{F}_{\text{out}} \subseteq \mathcal{F}_{\text{in}}\)).

In Proposition 4, \(U^\lambda(\mathcal{F}_{\text{in}})\) and \(V^\lambda_1(\mathcal{F}_{\text{in}})\) are determined by \(\mathcal{F}_{\text{in}}\) and \(\lambda\); they do not depend on \(\sigma\). For any \(C\) in \(V^\lambda_1(\mathcal{F}_{\text{in}})\), the difference in level between \(E\) and the leaf node of \(\text{Path}_\sigma(C, T_{\text{in}})\) — i.e., the value of \(\ell_{\text{out}}(\text{lastLeaf}_\sigma(C, T_{\text{in}})) - \ell_{\text{out}}(C) = \ell_{\text{in}}(\text{lastLeaf}_\sigma(C, T_{\text{in}})) - \ell_{\text{in}}(C) = \text{depth}_{\mathcal{F}_{\text{in}}}(C)\) — also does not depend on \(\sigma\). So even though the sets \(\text{Path}_\sigma(C, T_{\text{in}})\) may depend on the leaf enumeration \(\sigma\), we see from Proposition 4 that \(\mathcal{F}_{\text{out}}\) is uniquely determined by \(\mathcal{F}_{\text{in}}\) and \(\lambda\) up to a level-preserving essential isomorphism.
3.2.4.3 Linear-Time Implementation of Simplification Step 2

In the rest of this sub-subsection is assumed that each FCTS \((T, \ell)\) is represented in such a way that the root of \(T\) can be found in \(O(1)\) time and that all of the following tasks can be done in \(O(1)\) time for any node \(C\) of \(T\):

- Create a clone of \(C\) and add it to another FCTS (as a new child of some specified node of the latter).
- Find the parent of \(C\) in \(T\), if \(C\) is not the root.
- Determine the value of \(\ell(C)\).
- Determine whether or not \(C\) is a leaf of \(T\).

We also assume that, for any non-leaf node \(C\) of \(T\), we can find all the children of \(C\) in \(O(|\text{Children}_T(C)|)\) time.

In the rest of this section we describe simple but efficient implementations of step 2 and of a variant of step 2.

Let \(\mathcal{F}_{in} = (T_{in}, \ell_{in})\) be some FCTS, and let \(\sigma\) be an \(\ell_{in}\)-increasing leaf enumeration of \(\text{Leaves}(T_{in})\) such that, whenever \(X\) and \(Y\) are leaves of \(T_{in}\), the answer to the question

\[
\text{Does } X \text{ occur later than } Y \text{ in } \sigma? \tag{3.14}
\]

can be determined in \(O(1)\) time even if \(\ell_{in}(X) = \ell_{in}(Y)\).

Our implementation of step 2 runs in \(O(|\text{Nodes}(T_{in})|)\) time, and does not require the actual creation of the sequence \(\sigma\): We allow \(\sigma\) to be implicitly defined by some function \(f : \text{Leaves}(T_{in}) \times \text{Leaves}(T_{in}) \to \{\text{Yes, No}\}\) such that the answer to (3.14) for any two leaves \(X\) and \(Y\) of \(T_{in}\) is \(f(X, Y)\) and this can be computed in \(O(1)\) time.\(^2\)

\(^2\)Note that no algorithm which actually creates the sequence \(\sigma\) that is defined by any such function \(f\) can run in \(O(|\text{Nodes}(T_{in})|)\) time in all cases, because any comparison sort must perform \(\Omega(n \log n)\) comparisons to sort a set of \(n\) items (here, leaves) in the worst case [76, Thm. 8.1].
For every $\lambda > 0$ let $\mathcal{F}_{\text{out},\lambda}$ be the FCTS that should result from pruning $\mathcal{F}_{\text{in}}$ by removing branches of length $\leq \lambda$ using the leaf enumeration $\sigma$. We now explain how $\mathcal{F}_{\text{out},\lambda}$ can be constructed in $O(|\text{Nodes}(T_{\text{in}})|)$ time.

For each non-leaf node $D$ of $T_{\text{in}}$, we define $\text{next}(D, T_{\text{in}})$ to be the child of $D$ in $\text{Path}_{\sigma}(D, T_{\text{in}})$ (i.e., the child of $D$ that is an ancestor of $\text{lastLeaf}_{\sigma}(D, T_{\text{in}})$); if $D$ is a leaf of $T_{\text{in}}$ then we define $\text{next}(D, T_{\text{in}}) = D$.

During a single postorder traversal $\text{next}(D, T_{\text{in}})$, $\text{lastLeaf}_{\sigma}(D, T_{\text{in}})$, and $\text{depth}_{\mathcal{F}_{\text{in}}}(D)$ can be computed for all nodes $D$ of $T_{\text{in}}$ in $\sum_{D \in \text{Nodes}(T_{\text{in}})} O(1 + |\text{Children}_{T_{\text{in}}}(D)|) = O(|\text{Nodes}(T_{\text{in}})|)$ time. Then, for any given node $C$ of $T_{\text{in}}$ it is easy to determine in $O(1)$ time whether $C$ belongs to $U_{\lambda}(\mathcal{F}_{\text{in}})$, $V_{\lambda}(\mathcal{F}_{\text{in}})$, or to neither of those sets, and it is easy to find all the nodes of $\text{Path}_{\sigma}(C, T_{\text{in}})$ by following a chain of $\text{next}(D, T_{\text{in}})$ nodes that starts with $D = C$.

Hence we can construct $\mathcal{F}_{\text{out},\lambda}$ in $O(|\text{Nodes}(T_{\text{in}})|)$ time, for any positive $\lambda$ that the user may specify, in the following way:

1. Clone $\text{root}(T_{\text{in}})$, and initialize the output FCTS (i.e., the FCTS that will be output when the algorithm terminates) to be an FCTS whose only node is the clone of $\text{root}(T_{\text{in}})$.

2. Do a preorder traversal of the subgraph of $T_{\text{in}}$ that is induced by the set of nodes $U_{\lambda}(\mathcal{F}_{\text{in}}) \cup V_{\lambda}(\mathcal{F}_{\text{in}})$. (This is the rooted tree that is derived from $T_{\text{in}}$ by ignoring all nodes which do not lie in the set $U_{\lambda}(\mathcal{F}_{\text{in}}) \cup V_{\lambda}(\mathcal{F}_{\text{in}})$. Note that this set contains $\text{root}(T_{\text{in}})$ and all the ancestors of each node in the set.) When any node $E$ is visited during the traversal, do the following:

   (2a) If $C \in U_{\lambda}(\mathcal{F}_{\text{in}}) \setminus \{\text{root}(T_{\text{in}})\}$, then create a clone of $C$ and add it to the output FCTS.

   (2b) If $C \in V_{\lambda}(\mathcal{F}_{\text{in}})$, then find all the nodes of $\text{Path}_{\sigma}(C, T_{\text{in}})$ and, for every such node $D$, create a clone of $D$ and add it to the output FCTS (unless
It is evident that $\mathcal{F}_{\text{out,}l}$ can be constructed in this way, since steps (2a) and (2b) will create clones of all nodes of types (i) and (ii) in Proposition 4 (except the root of $T_{\text{in}}$) and add them to the output FCTS.

Step 3 of $(\lambda, k)$-simplification simplifies $\mathcal{F}^{\text{crit}}$, where $\mathcal{F}$ is the output of step 2. We can construct $\mathcal{F}^{\text{crit}}$ directly, without constructing $\mathcal{F}_{\text{out,}l}$, using a modified version of the algorithm described above in which (2a) and (2b) are replaced with:

(2a’) If $C \in \mathbf{U}^{\lambda}(\mathcal{F}_{\text{in}}) \setminus \{\text{root}(T_{\text{in}})\}$, and $\text{Children}_{T_{\text{in}}}(C)$ contains two or more nodes in $\mathbf{U}^{\lambda}(\mathcal{F}_{\text{in}}) \cup \mathbf{V}^{\lambda}_{1}(\mathcal{F}_{\text{in}})$, then create a clone of $C$ and add it to the output FCTS.

(2b’) If $C \in \mathbf{V}^{\lambda}_{1}(\mathcal{F}_{\text{in}})$, then create a clone of the node $\text{lastLeaf}_{\sigma}(C, T_{\text{in}})$ and add it to the output FCTS.

Here (2b’) assumes that $T_{\text{in}}$ has at least two nodes.

3.2.5 Elimination of Internal Edges of Length $\leq \lambda$ from $\mathcal{F}^{\text{crit}}$

Step 3 of $(\lambda, k)$-simplification is to eliminate internal edges of length $\leq \lambda$ from $\mathcal{F}^{\text{crit}}$, where $\mathcal{F}$ is the FCTS that results from step 2 of $(\lambda, k)$-simplification. We now mathematically specify the output of step 3, and then present an algorithm which implements step 3.

3.2.5.1 Specification of Simplification Step 3

Let $\mathcal{F} = (T, \ell)$ be any FCTS. Then, for each $\lambda > 0$, the result of eliminating internal edges of length $\leq \lambda$ from $\mathcal{F}^{\text{crit}}$ is the FCTS $\mathcal{F}^{\text{crit}}(\lambda)$ that we will define below. The
definition will use some notation which we now introduce.

The set \( \{ \ell(C) - \ell(D) \mid C, D \in \text{Crit}(\mathcal{G}) \setminus \text{Leaves}(\mathcal{G}) \text{ and } D \in C_{\downarrow} \} \) will be denoted by \( \mathcal{D}(\mathcal{G}) \), and \( d_1^\mathcal{G} < d_2^\mathcal{G} < \cdots < d_{|\mathcal{D}(\mathcal{G})|}^\mathcal{G} \) will denote the elements of \( \mathcal{D}(\mathcal{G}) \) in ascending order. (Note that all elements of \( \mathcal{D}(\mathcal{G}) \) are positive.) We define \( d_0^\mathcal{G} = 0 \).

For any \( \lambda > 0 \), we define \( \text{pred}_{\mathcal{G}}(\lambda) = \max\{d \in \mathcal{D}(\mathcal{G}) \cup \{0\} \mid d < \lambda\} \).

**Example 5.** Let \( \mathcal{G} \) be the FCTS shown in Figure 3.18. Then we see from Figure 3.18 that \( \text{Crit}(\mathcal{G}) \setminus \text{Leaves}(\mathcal{G}) = \{v_1, v_4, v_5, v_{15}, v_{16}\} \) and \( D(\mathcal{G}) = \{1, 5, 6, 7, 11, 12\} \).

It follows, for example, that, \( d_1^\mathcal{G} = 1, d_2^\mathcal{G} = 5 \), and \( \text{pred}_{\mathcal{G}}(\lambda) = 1 \) for \( 1 \leq \lambda < 5 \).

Now we define \( \mathcal{G}^{\text{crit}}(0) = \mathcal{G}^{\text{crit}} \) and, for all \( \lambda > 0 \), we recursively define \( \mathcal{G}^{\text{crit}}(\lambda) \) to be the FCTS that has the following five properties:

- **E1:** \( \mathcal{G}^{\text{crit}}(\lambda) \subseteq \mathcal{G}^{\text{crit}} \)
- **E2:** \( \text{LCN}(\mathcal{G}^{\text{crit}}(\lambda)) = \text{LCN}(\mathcal{G}) \)
- **E3:** \( \text{Leaves}(\mathcal{G}^{\text{crit}}(\lambda)) = \text{Leaves}(\mathcal{G}) \)
- **E4:** If \( \lambda \notin \mathcal{D}(\mathcal{G}) \), then \( \mathcal{G}^{\text{crit}}(\lambda) = \mathcal{G}^{\text{crit}}(\text{pred}_{\mathcal{G}}(\lambda)) \).
- **E5:** For every \( C \in \text{Nodes}(\mathcal{G}^{\text{crit}}) \setminus (\text{Leaves}(\mathcal{G}) \cup \{\text{LCN}(\mathcal{G})\} \cup \{\text{root}(\mathcal{G})\}) \) and every \( i \in \{0, \ldots, |\mathcal{D}(\mathcal{G})| - 1\} \), we have that \( C \in \text{Nodes}(\mathcal{G}^{\text{crit}}(d_{i+1}^\mathcal{G})) \) if, and only if, \( C \in \text{Nodes}(\mathcal{G}^{\text{crit}}(d_i^\mathcal{G})) \) and \( \ell(C) - \ell(\text{parent}_{\mathcal{G}^{\text{crit}}(d_i^\mathcal{G})}(C)) > d_{i+1}^\mathcal{G} \).

E1 implies that \( \text{Nodes}(\mathcal{G}^{\text{crit}}(\lambda)) \subseteq \text{Nodes}(\mathcal{G}^{\text{crit}}) = \text{Crit}(\mathcal{G}) \cup \{\text{root}(\mathcal{G})\} \), and also implies that \( \text{root}(\mathcal{G}^{\text{crit}}(\lambda)) = \text{root}(\mathcal{G}) \).

**Example 6.** Figure 3.19 shows the FCTS \( \mathcal{G}^{\text{crit}}(\lambda) \) in the case where \( \mathcal{G} \) is the FCTS that is shown in Figure 3.18 and \( 1 \leq \lambda < 5 \). Here \( d_1^\mathcal{G} = 1 \) and \( d_2^\mathcal{G} = 5 \) (as we observed in Example 5). Since \( d_1^\mathcal{G} \leq \lambda < d_2^\mathcal{G} \), it follows from E4 that \( \mathcal{G}^{\text{crit}}(\lambda) = \mathcal{G}^{\text{crit}}(d_1^\mathcal{G}) = \mathcal{G}^{\text{crit}}(1) \). The node \( v_{16} \) in Figure 3.18 is not a node of \( \mathcal{G}^{\text{crit}}(d_1^\mathcal{G}) \); indeed, when we put \( i = 0 \) and \( c = v_{16} \), the condition \( \ell(C) - \ell(\text{parent}_{\mathcal{G}^{\text{crit}}(d_i^\mathcal{G})}(C)) > d_{i+1}^\mathcal{G} \) in E5 is
Figure 3.19: The Effect of Eliminating Internal Edges of Length $\leq \lambda$ from an FCTS.

The effect of eliminating internal edges of length $\leq \lambda$ from $\mathcal{F}^{\text{crit}} = (T^{\text{crit}}, \ell^{\text{crit}})$ is shown here, in the case where $\mathcal{F} = (T, \ell)$ is the FCTS of Figure 3.18 and $1 \leq \lambda < 5$. The nodes and edges of the resulting FCTS $\mathcal{F}^{\text{crit}}(\lambda)$ are shown as fat/thick black nodes and edges. Other nodes and edges of the tree $T$ of Figure 3.18 are colored gray, but two of those nodes ($v_9$ and $v_{11}$ in Figure 3.18) and three of those edges are partially or completely hidden by the thick black edge that joins $v_4$ to $v_{17}$. Note that, since 2 is a possible value of $\lambda$ in this figure, and since $\mathcal{F}$ is the result of applying steps 1 and 2 of $(\lambda, k)$-simplification to the FCTS shown in Figure 3.14 in the case $\lambda = 2$ and $k = 1$, the FCTS shown in this figure is the $(2, 1)$-simplification of the FCTS shown in Figure 3.14.

not met since $\text{parent}_{\mathcal{F}^{\text{crit}}(d_{16})}(v_{16}) = v_{15}$ and $\ell(v_{16}) - \ell(v_{15}) = 1 = d_{15}^{-\mathcal{F}}$. But $E1 – E5$ imply that the other 12 nodes of $\mathcal{F}^{\text{crit}}$ are nodes of $\mathcal{F}^{\text{crit}}(\lambda)$.

3.2.5.2 Implementation of Simplification Step 3

It is possible to perform simplification step 3 (i.e., to construct $\mathcal{F}^{\text{crit}}(\lambda)$ from $\mathcal{F}^{\text{crit}}$) by direct application of $E1 – E5$. However, this would require computation of the sorted sequence $d_{1}^{-\mathcal{F}} < d_{2}^{-\mathcal{F}} < \cdots < d_{k}^{-\mathcal{F}}$, where $d_{k}^{-\mathcal{F}}$ is $\lambda$ or $\text{pred}_{\mathcal{F}}(\lambda)$ according to whether $\lambda \in D(\mathcal{F})$ or $\lambda \notin D(\mathcal{F})$, followed by $k$ tree traversals that successively find the nodes of $\mathcal{F}^{\text{crit}}(d_{1}^{-\mathcal{F}}), \mathcal{F}^{\text{crit}}(d_{2}^{-\mathcal{F}}), \ldots, \mathcal{F}^{\text{crit}}(d_{k}^{-\mathcal{F}})$.
Algorithm 1 below, which performs just one tree traversal after the initial cloning step, will usually be a much more efficient implementation of step 3. It inputs a FCTS $\mathcal{F}_{\text{in}} = (T_{\text{in}}, \ell_{\text{in}})$ and a positive $\lambda$, and constructs $\mathcal{F}_{\text{in}}^{\text{crit}}(\lambda)$ by creating a clone $(T, \ell)$ of $\mathcal{F}_{\text{in}}^{\text{crit}} = (T_{\text{in}}^{\text{crit}}, \ell_{\text{in}}^{\text{crit}})$ and then labeling each node $C$ of $T$ with a value $C.\text{label}$ such that $\mathcal{F}_{\text{in}}^{\text{crit}}(\lambda) = (T, \ell) - \{C \in \text{Nodes}(T) \mid C.\text{label} \leq \lambda\}$. The correctness of this algorithm is proved in Appendix A.

**Algorithm 1:** Eliminate Internal Edges of Length $\leq \lambda$ from $\mathcal{F}_{\text{crit}}$.

**inputs:** a FCTS $\mathcal{F}_{\text{in}} = (T_{\text{in}}, \ell_{\text{in}})$; a positive real value $\lambda$

**output:** a FCTS $\mathcal{F}_{\text{out}}$ that satisfies $\mathcal{F}_{\text{out}} \subseteq \mathcal{F}_{\text{in}}$

$(T, \ell) \leftarrow$ a clone of $(T_{\text{in}}^{\text{crit}}, \ell_{\text{in}}^{\text{crit}})$;

root$(T).\text{label} \leftarrow \infty$;

LCN($T$).label $\leftarrow \infty$;

foreach $C \in \text{Children}_T(LCN(T))$ do labelDescendants$(C, T, \ell, \lambda)$;

$\mathcal{F}_{\text{out}} \leftarrow (T, \ell) - \{D \in \text{Nodes}(T) \mid D.\text{label} \leq \lambda\}$;

**Procedure** labelDescendants$(C, T, \ell, \lambda)$

if $C \notin \text{Leaves}(T)$ then

$C' \leftarrow C$;

repeat

$C' \leftarrow \text{parent}_T(C)$;

$C.\text{label} \leftarrow \ell(C) - \ell(C')$;

until $(C.\text{label} > \lambda$ or $C.\text{label} \leq C'.\text{label})$;

foreach $D \in \text{Children}_T(C)$ do labelDescendants$(D, T, \ell, \lambda)$;

else $C.\text{label} \leftarrow \infty$;

If we write $h(\mathcal{F}_{\text{in}}, \lambda)$ to denote the length $l \geq 1$ of the longest chain of nodes $C \triangleright_T \cdots \triangleright_T C_i$ in $\text{Crit}(\mathcal{F}_{\text{in}})$ for which $\ell_{\text{in}}(C_1) - \ell_{\text{in}}(C_i) \leq \lambda$, then we see from the
initial step “(T, ℓ) ← a clone of \((T_{in}^{\text{crit}}, \ell_{in}^{\text{crit}})\)” of Algorithm 1 and from the repeat ... until loop in labelDescendants that, under the assumptions which are stated at the beginning of Sub-subsection 3.2.4.3, the running time of Algorithm 1 is \(O(|\text{Nodes}(\mathcal{F}_{in})| + h(\mathcal{F}_{in}, \lambda)|\text{Crit}(\mathcal{F}_{in})|).\)

### 3.2.6 A Potential Application of FCTSs

As an example of a potential biological application, an experiment applying FCTS to identify the difference of two very similar, but not identical, macromolecules is used. These data sets originate from the work of San Martín et al. [77], which investigated some biological questions associated with adenoviruses. The data sets were kindly provided by Roberto Marabini of the Universidad Autónoma de Madrid.

The macromolecule has an icosahedral shape with a diameter of approximately 900 Å. At each of the 12 vertices of the icosahedron, there is a substructure referred to as a *penton*, and the rest of the surface of the icosahedron consists of 240 *hexons*. To reflect this, the simplified FCTSs of these viruses would be expected to have 252 leaves, one for each penton or hexon. This is indeed the case, as will be seen.

In the course of their work, San Martín et al. [77] produced a *mutant* version of the *wildtype* version of the adenovirus they were investigating. The two are identical except for a change in a protein (called IIIa). Surface renderings and central cross-sections of the two versions are shown in Figure 3.20. As will be now described, in spite of their great similarity, the two versions can be distinguished from each other by an obvious topological difference between their simplified FCTSs.

Each version of the virus studied by San Martín et al. [77] was represented by a grayscale volume on a 275 × 275 × 275 array of sample points. For each digital picture \((V, \pi, f)\), a \(\text{FCTS}_{(V, \pi, f)}\) was constructed using 6-adjacency for the adjacency relation \(\pi\), and then a \((\lambda, k)\)-simplification of \(\text{FCTS}_{(V, \pi, f)}\) for various choices of \(\lambda\) and \(k\) was computed.
It was found that $\lambda = 10$ and $k = 799$ were good choices that yielded topologically different simplified FCTSs for the two versions of the virus. These simplified FCTSs are shown in Figure 3.21. Each simplified FCTS has 252 leaves, corresponding to the 12 pentons and 240 hexons. For the wildtype version, the lowest critical node is the parent of all 252 leaves; see Figure 3.21(a). For the mutant version, the lowest critical node is the parent of the 12 leaves that correspond to pentons, but is the grandparent of the 240 leaves that correspond to hexons; see Figure 3.21(b). These simplified FCTSs indicate that for the mutant version of the virus, there is a substantial range of threshold levels (such as level A in Figure 3.21(b)) at which the pentons are disconnected from each other and from the hexons, but the hexons are connected to each other; for the wildtype version there is no such range of threshold values. Interestingly, San Martín et al. [77] do not mention this difference between the two versions of the virus, although they do point out that in images of the mutant version pentons have lower graylevels than hexons. (The latter can be seen in Figure 3.20(d), and is also indicated by Figure 3.21(b); when the image of the mutant virus is thresholded at the graylevel B in Figure 3.21(b), hexons are observable but pentons are not.)

The simplified FCTSs may possibly have revealed a previously unobserved difference between the mutant and the wildtype versions of the virus: for the mutant
Figure 3.21: \((\lambda, k)\)-simplifications of FCTSs of Wildtype and Mutant Adenoviruses.

\((\lambda, k)\)-simplifications of FCTSs of wildtype (a) and mutant (b) adenoviruses, where \(\lambda = 10\) and \(k = 799\). In (a), the lowest critical node (represented by the horizontal line segment) is the parent of all 252 leaves of the tree. In (b), the lowest critical node (represented by the horizontal line segment above line A) is the parent of the rightmost 12 leaves, which correspond to pentons, but is the grandparent of the other 240 leaves, which correspond to hexons.

version, there is a substantial range of threshold values at which the hexons are connected to each other, but no penton is connected to a hexon or to another penton. To investigate whether this is a genuine difference between the two versions of the virus or merely a difference between the specific volume images from which FCTSs were produced, we carried out a further study.

Ideally, we would have compared simplified FCTSs of, say, 10 independently reconstructed volume images of each version, but such data were not available to us. To validate the results presented in [20], the following approach was done. For each version of the virus, we randomly selected 2000 out of 3000 available projection images, and used them to reconstruct a volume image on a \(275 \times 275 \times 275\) array of points. This was repeated 10 times.

For each of the 20 resulting volume images, we produced a simplified FCTS using the above-mentioned parameters. In each of the 10 simplified FCTSs of the mutant adenovirus, the lowest critical node had 13 children, 12 corresponding to the pentons and the 13\(^{th}\) being the root of a subtree whose leaves corresponded to
the hexons, as in Figure 3.21(b). But this was not true of the 10 simplified FCTSs of the wildtype adenovirus; they were all similar to Figure 3.21(a).

These results provide some evidence to support the hypothesis that images of the mutant version of the virus can be distinguished from images of the wildtype version by the existence in the former (but not the latter) of a substantial range of threshold values with the above-mentioned properties. However, more investigation would be needed to confirm this hypothesis. In any event, this example illustrates how our simplified FCTSs may reveal interesting structural differences between two similar macromolecules.
Chapter 4

Using Component Trees to Explore Biological Structures

This chapter describes some properties of component trees that can be used to explore biological structures. The first component tree property discussed here is the bijective map between a digital picture \((V, \pi, f)\) and the component tree \(T_{(V, \pi, f)}\). The component tree creates a visual representation for a digital picture in which each node represents a unique \(\pi\)-connected component in the digital picture. This bijective map can provide a useful visualization tool to explore the biomolecular structure and can be helpful to guide macromolecular docking.

An interactive selection of a subtree in \(T_{(V, \pi, f)}\) for example, can produce a segmentation of \((V, \pi, f)\) that corresponds to a unique node of \(T_{(V, \pi, f)}\). Conversely, the selection of a \(\pi\)-connected component \(C \in \mathcal{C}_{(V, \pi, f)}\) can highlight a node or a subtree of \(T_{(V, \pi, f)}\). In addition, the effect of any operation on a digital picture (or a component tree) may be visualized in the corresponding component tree (or a digital picture). For example, if a component tree \(T_{(V, \pi, f)}\) is simplified by the methodology described in Subsection 3.2.2, the effect of this simplification may change the intensities of the elements of \(V\) in the digital picture \((V, \pi, f)\). On the other
hand, if a smooth filter is applied to the digital picture \((V, \pi, f)\), the structure of the component tree \(T_{(V, \pi, f)}\) may also change.

The second property is the hierarchical relationship between the connected components of a digital picture. Recall that if \(D \in \mathcal{C}_{(V, \pi, f)}\) is a subset both of \(C_1 \in \mathcal{C}_{(V, \pi, f)}\) and of \(C_2 \in \mathcal{C}_{(V, \pi, f)}\), then either \(C_1\) or \(C_2\) must be a subset of the other. This hierarchical relationship, for example, allows a visual exploration of the elements of \(\mathcal{C}_{(V, \pi, f)}\) in the component tree \(T_{(V, \pi, f)}\) that belong to the same region or subtree.

As explained in Subsection 3.1.3, a component tree is a rooted tree that contains information regarding the connected components that are obtained when a digital picture is thresholded at different levels. More specifically, each node of a digital picture’s component tree is a connected component of the picture elements whose intensities are greater than or equal to the level of that node, and the ancestor-descendant relationships between nodes correspond to the inclusion relationships between these connected components.

The results presented in Section 3.1 provide the mathematical background for developing interactive visualization tools based on component trees that can be used to investigate biomolecular structures. We believe that such visualization tools will be of value to scientists and professionals who study and work with these structures.

In the current chapter, we present various ways of using component trees to explore macromolecular information. The first section introduces visualization tools developed in the course of this research. This set of tools, named MataExplor, is a collection of programs that compute, simplify, plot, and manipulate component trees in an interactive environment. The second section presents various scenarios for using a component tree to explore a macromolecular complex by manual or automated selection of nodes. The last two sections present two case-studies: in them component trees are used to investigate (i) the proteins of the procapsid of the bacteriophage \(\phi 6\) and (ii) to indicate the possible structure of an protein.
4.1 **MataExplor: An Interactive Tool for Component Tree Exploration**

A component tree is a topological and geometrical descriptor that creates a graphical representation for the relationship between connected components of a finite set. More specifically for this dissertation, component trees offer an alternative way to explore biological data by the creation of a compact tree-like data structure that captures the essential structural information for the data. For example, the selection of nodes or subtrees of a component tree for a digital picture that represents a macromolecular complex can be used to identify parts or subunits of the whole complex.

Over the years, component trees have been applied to various types of image processing methods such as image filtering and segmentation [68, 71, 78, 79], image compression [80], image retrieval [81] and object identification [82]. Component trees have been applied to biomedical images in order to segment 3D magnetic resonance imaging (MRI) and magnetic resonance angiograms (MRA). In the work of Dokladal *et al.* [83] and Passat *et al.* [84], component trees were used to segment brain structures in MRI images while in the work of Wilkinson *et al.* [85], component trees were used to segment vessels in MRA brain images. In this dissertation, we develop an interactive tool, which we call MataExplor, to explore component trees of digital pictures.

*MataExplor* is a graphical user interface (GUI)-based tool for the exploration of component trees of digital pictures, which combines various programs to: (i) compute, simplify and plot component trees, (ii) convert different file types, and (iii) save volumes that correspond to connected components of a digital picture. Under continuing development, MataExplor is designed to provide a simple and intuitive environment where the user can apply component trees to understand the structural
The six modules that compose MataExplor are File Format Conversion, Component Tree Computation, Component Tree Simplification, Component Tree Explore, Save Component Tree, and Save Connected Components. After the files are converted to the appropriate format in the File Format Conversion module, the component trees are computed and simplified in the Component Tree Computation and Simplification modules respectively. Then, the component trees can be plotted and manipulated using the Component Tree Explore module. There is also the option to save the trees in PDF or JPEG format using the Save Component Tree module and to save particular connected components though the Save Connected Components module.

The current implementation of MataExplor can accept three different file formats as input: voronoi, spider [86], and map [25]. The voronoi file format is the default input format and is based on the DIGFile library of the software package SNARK09 [87, 88]. This library provides routines that can be easily used to access, extract, and modify the data stored in the voronoi files (for more details about the voronoi file, see [89] and http://www.dig.cs.gc.cuny.edu/manuals/schemas/DIG.xsd that provides a skeleton of an XML Schema describing a DIG data file). The spider and map formats must be converted to a voronoi file in order to be usable in MataExplor. This conversion can be done with the File Format Conversion module.

In order to compute a component tree using MataExplor, the user must first open a volume by clicking the icon “+” shown in the area indicated by A-1 in Figure 4.2.
The region indicated by A shows the options of the Explore widget: 1- open a new volume file; 2- tree zoom in; 3- tree zoom out; 4- save a volume with all the voxels of a selected node (connected component); 5- show number of voxels of a component. B lists all the volumes available for computing a component tree and C lists all component trees previously computed for a selected volume. This screenshot shows a component tree for the emb_1649 volume using 255 bins and its two simplifications: \((\lambda = 1,k = 5)\) and \((\lambda = 2,k = 20)\). D shows the parameters to compute and simplify a component tree. E lists all volumes created by the use of option 4 in the Explore widget and F indicates the Display widget.
After opening the volume, MataExplor will load it into the computer memory and then insert it into the Volume List widget indicated by the letter B in the same Figure 4.2. This list contains all the available volumes that can be used to compute a component tree. In this particular case, only one volume is listed in this widget: emd_1649.

The computation of component trees in MataExplor is based on the use of threshold levels of a digital picture as was described in Subsection 3.1.3. The user has the option to use all intensity levels presented in a digital picture or to quantify these intensity levels into a discrete set of equally spaced values called bins. The use of bins to create component trees can reduce the computation time and therefore provides a faster overview of the structures captured by the component trees. As will been seen in our experiments reported in Sections 4.2 and 4.3, the use of 256 bins provides sufficient information for analyzing the investigated digital pictures.

By selecting a volume in the Volume List widget, the user can start the process of computing and simplifying component trees by entering the parameters Bins, λ, and k, and then clicking the button “run.” These parameters and the button “run” can be accessed in the area labeled with the letter D in the Component Tree widget depicted in Figure 4.2. The default value for the parameter Bins is equal to the number of intensity levels presented in the volume being used. The area indicated by the letter C in this same figure provides a list of all the previously computed and simplified component trees.

After computing the component trees, they are visualized in the Display widget and/or exported in either JPEG or PDF format. Our component trees are plotted using the phenogram-like [90, 91] rooted tree diagram and are exported to the Newick [92] format using the Drawgram tree-plotting program available in the Phylip package [93, 94]. As was explained in Subsection 3.1.5, there are four kinds of nodes in our phenogram-like rooted trees:
1. The root, which is represented by the upper endpoint of the uppermost vertical line.

2. Nodes that are not the parent node of any edge (usually referred to as leaves); whenever a vertical line’s lower endpoint does not lie on a horizontal line, that lower endpoint represents a leaf.

3. Nodes that are the parent nodes of more than one edge; these are represented by horizontal lines.

4. Nodes other than the root that are the parent node of exactly one edge; these are not explicitly represented but correspond to points that occur somewhere in the interior of a vertical line.

An example of a phenogram-like rooted tree is shown in the MataExplore screenshot depicted in Figure 4.2.

Component trees can be interactively manipulated in MataExplore using options such as zoom out/in, adding or removing label nodes, and dragging nodes or edges and thereby change the way the tree was originally plotted. By selecting a node in a component tree, the user has the option to save a volume containing all the voxels that belong to the component associated with the selected node. MataExplore can also render the surface of the saved volume through a command line integration with Chimera [25].

Figure 4.3 shows a zoomed part of the component tree presented in Figure 4.2, with the label of each node displayed inside of the tan circle. It is important to remember that the label of a node corresponds to the intensity level used to create the connected component associated with that node. For example, the label of the left-most node in Figure 4.3 is 62. This indicates that all the voxels that belong to that node (which is a connected component) have intensity level greater than 62.
Figure 4.3: Zooming a Component Tree Using MataExplor.
The component tree display widget of MataExplor shows a zoomed middle-left part of the component tree presented in Figure 4.3. This time the labels of the component tree are shown in the tan circles. By selecting a node and using option 5 of the component tree explore widget, the user can see the number of voxels in a node. The dash-outlined leaf indicates the selected node.

Additionally, MataExplor can display information about the number of voxels of a connected component. By positioning the mouse over a tan circle of a node, the number of voxels in the component associated with that node is displayed. For example, the number 26 shown at the bottom-left of the Display widget in Figure 4.3 indicates the number of voxels in the connected component associated with the node that has the dash-outlined tan circle.

Once a component tree is computed, several simplifications can be applied to it by the selection of the simplification parameters $\lambda$ and $k$. All the simplifications for a given component tree are stored and listed in the MataExplor panel enabling the user to browse through all the simplifications by selecting a simplified tree in the list of previously computed component trees.

Internally, the MataExplor architecture was designed on top of a general-purpose
C++ class library and the framework Qt version 4.8 [95]. MataExplor has been tested for Linux platforms and will soon be available for download upon request.

4.2 Examples of Macromolecular Exploration Using Component Trees

To illustrate how a tool of this kind can be used, we now describe three different scenarios in which regions in a density map can be identified by the manual or automatic selection of nodes in a component tree. In these examples, a density map of the microtubule binding patterns of dimeric kinesins [96] (EMDB access code 1032 [96]) with $100 \times 100 \times 100$ voxels and a voxel spacing of 5.68 Å is used. This will be referred to as the EMDB-1032 density map from now on. The structure of this macromolecular complex is composed of two different kinds of sub-structures: microtubules and dimer kinesins. The microtubules are long hollow cylinders made up of protofilaments (polymerised $\alpha$- and $\beta$-tubulin dimers). The lateral association of 15 protofilaments generates the microtubule, a cylindrical structure with imperfect helical symmetry. Dimer kinesins are motor proteins that move along the microtubule. The kinesins are attached to the microtubule by a binding site—a region on a protein with which specific other molecules and ions form a chemical bond. Figure 4.4 shows a surface rendering and a slice of the microtubule binding patterns of dimeric kinesins.

4.2.1 Interactive Component Tree Exploration

The first scenario presented here shows how connected components of a digital picture can be identified by a manual selection of nodes in a component tree. For this we use a $(0, 20)$-simplification of a component tree of the EMDB-1032 density map. Figure 4.5(a) shows such a $(0, 20)$-simplified component tree in which a
Figure 4.4: Microtubule Binding Patterns of Dimeric Kinesins.
(a) Surface rendering and (b) central slice of a density map of the microtubule binding pattern of dimeric kinesins (density map with EMDB access code 1032 [96]).

For this illustration, we selected six nodes in the component tree presented in Figure 4.5(a). The positions of these nodes in the magnified subtree are indicated by the colored arrows. Recall that each node is a connected component of the voxels whose intensities (in the EMDB-1032 density map) are greater than or equal to the level of that node. Take the case of the node indicated by the red arrow in Figure 4.5(a). This component is represented by the red surface rendering in Figure 4.5(b). It contains one of the 15 vertical sections of the microtubule and four dimer kinesins. The nodes indicated by the blue, green, yellow, and pink arrows are represented by the blue, green, yellow, and pink kinesins in Figure 4.5(c). Figure 4.5(d) shows the above-mentioned components in their positions within the density map.

It is important to note that in Figure 4.5, the red component contains the other four segmented components (blue, green, yellow, and pink). This reflects the fact that in the component tree, the node indicated by the red arrow is an ancestor of the nodes indicated by the blue, green, yellow, and pink arrows. The node indicated by the cyan arrow in Figure 4.5(a) is another descendant of the node indicated by the red arrow. This component lies in the vertical section of the microtubule to which the kinesins are attached in Figure 4.5(b); it is shown as the cyan segment in Figure
Figure 4.5: Interactive Component Tree Exploration.
(a) The (0,20)-simplified component tree constructed from the EMDB-1032 density map. (b) Surface rendering of the component (node) indicated in (a) by the red arrow. This component contains one of the 15 vertical sections of the microtubule and four dimer kinesins. (c) The four kinesins (the components indicated in (a) by the blue, green, pink, and yellow arrows). (d) Positions in the density map of these five components. (e) Relationship between these five components and a sixth component that is indicated by the cyan arrow in (a).

4.2.2 Component Tree Exploration by Threshold Level

The second scenario illustrates how a component tree can be used to explore the components of a digital picture at various threshold levels. Recall that the level of any component tree node is defined to be the minimum of the intensity levels of the picture elements in that component. For any positive real number $\tau$, we
threshold a component tree at the level $\tau$ by omitting all the nodes of level less than $\tau$. When we omit those nodes from the tree, we are left with a forest of subtrees. Our visualization tool can display surface renderings of those subtrees. (By a surface rendering of a subtree we mean a surface rendering of the node/component which is the root of that subtree, and which therefore contains all the other nodes of the subtree.)

Figure 4.6 shows surface renderings of the subtrees obtained by thresholding a component tree at three different threshold levels $\tau_1 < \tau_2 < \tau_3$. The component tree that is thresholded in this example is a $(8,50)$-simplification of a component tree of the EMDB-1032 density map.

Figure 4.6(a) shows that thresholding at level $\tau_1$ produces a forest of 17 subtrees. Fifteen of these subtrees have five or six leaves; these subtrees represent structures, each of which comprises a vertical section of the microtubule and four or five attached kinesins. The other two subtrees have just one leaf each; these represent the two kinesins indicated by the cyan arrows in Figure 4.6(a). When the EMDB-1032 density map is thresholded at level $\tau_1$, those two kinesins are not connected (within this density map) to the rest of the macromolecule. In Figures 4.6(b) and (c), which show surface renderings of the subtrees obtained by thresholding at levels $\tau_2$ and $\tau_3$, respectively, the red and blue rectangles highlight surface renderings of those parts of the red and blue subtrees that lie on or below the green line. In Figure 4.6(c), the purple circle highlights a component which corresponds to a tree node that is indicated by the purple arrow.

As the threshold level is gradually lowered from the highest to the lowest level that occurs in the density map, the visualization tool will show surface renderings of components as they come into existence or merge with other components. For example, when the threshold level falls to the level of the closest common ancestor of the six leaves of the red subtree in Figure 4.6(b), the components that represent
Figure 4.6: Component Tree Exploration by Threshold Level. The (8,50)-simplified component tree of the EMDB-1032 density map is shown with three different threshold levels $\tau_1$, $\tau_2$, and $\tau_3$ in panels (a), (b), and (c). The right side of each panel shows surface renderings of components that are the roots of the forest produced when the tree is thresholded at the level indicated in that panel.

The five kinesins and the microtubule in the red rectangle in Figure 4.6(b) will merge into a single component that represents one of the 15 vertical sections shown in Figure 4.6(a). Users can explore the component tree by interactively raising and lowering the threshold, and by doing so, gain a better understanding of the structural relationships between components.
Figure 4.7: Automatic Component Tree Exploration.
(a) The part of the component tree (top left) that is highlighted by a rectangle. The 22 small red disks indicate the nodes in the component tree that are associated with components whose component volumes are greater or equal to 420,000 Å³ and are less than or equal to 450,000 Å³. (b) The various components associated with these nodes. Reproduced from [21].

4.2.3 Automatic Component Tree Exploration

The last scenario presented here shows how a visualization tool can automatically find all the tree nodes that represent components for which the value of a certain attribute of interest falls within a user-specified range, and then display those components. We will illustrate this using the tree in Figure 4.7(a), and using each component volume (which we define in the next paragraph) for each connected component as its attribute of interest.

Let \((V, π, f)\) be a digital picture based on voxels, let \((N_{(V,π,f)}, E_{(V,π,f)})\) be its component tree, and let \(ω\) denote the volume of a single element (i.e., the cube of the voxel spacing); this is 193.10 Å³ for the EMDB-1032 density map we are using in this section. For any node \(C ∈ N_{(V,π,f)}\), we define the component volume of \(C\) as

\[
\text{volume}(C) = \text{card}(C)ω
\] (4.1)
where \( \text{card}(C) \) is the cardinality of the component \( C \) (i.e., the number of voxels in \( C \)). For any component tree and any two positive real numbers \( w \) and \( z \) such that \( w \leq z \), our visualization tool can produce a surface rendering of all the nodes \( C \) such that \( w \leq \text{volume}(C) \leq z \).

Figure 4.7 shows an example of this for the density map used in the previous scenarios. For this example, the component tree was simplified using \( k = 30 \) and \( \lambda = 2 \), and the parameters \( w \) and \( z \) were selected to be 420,000 Å\(^3\) and 450,000 Å\(^3\). The 22 nodes shown as small red disks in the magnified part of Figure 4.7(a) are the nodes that have a component volume between 420,000 Å\(^3\) and 450,000 Å\(^3\). Figure 4.7(b) shows a surface rendering of these 22 components. Note that only 16 components can be seen in Figure 4.7(b). This reflects the fact that the 22 nodes include six parent-child pairs; the six components that are the child nodes in these pairs are not visible because they are contained in the components that are their parents.

4.3 Case Study 1: Using Component Trees to Identify Proteins in the Procapsid of the Bacteriophage \( \phi 6 \)

A component tree is a topological and geometric descriptor that captures information regarding structure in an image based on the connected components determined by various grayness thresholds. We believe that the interactive visual exploration of component trees of (the density maps of) macromolecular complexes can yield much information about their structure. To illustrate how component trees can convey important structural information, we consider component trees of four recombinant mutants of the procapsid of a bacteriophage (cystovirus \( \phi 6 \)) and show how
The differences between the component trees reflect the fact that each non-wild-type mutant of the procapsid has an incomplete set of constituent proteins.

In this section, we present our initial findings from an interactive exploration of the component trees for procapsids of four versions using MataExplor. Our findings suggest that suitably simplified component trees can be used to distinguish the four versions of a bacteriophage. Furthermore, this experiment reinforces the idea that component trees may well be useful for the identification of substructures in a macromolecular complex.

The bacteriophage φ6 is a virus containing a genome of three segments of double-stranded RNA. The RNA packaging, replication, and transcription mechanisms of φ6 are very similar to those of reoviruses that contain species infectious to many animals, making the species an excellent model system for these important pathogens [97]. The initial step in the φ6 replication is the assembly of a closed and unexpanded procapsid (PC) that is responsible for viral RNA packaging, transcription, and genome replication.

The PC has dodecahedral morphology with deeply recessed vertices; its diameter is approximately 860 Å. For wild-type (i.e., containing all its native protein elements) specimens of φ6, the PC is comprised of four proteins: P1, P2, P4, and P7. P1 is the structural protein responsible for the overall structure of the PC, which consists of 120 copies of this protein in two different conformations: P1a and P1b. The first, P1a, creates pentamers that are centered on the 12 faces of the PC, which therefore contains a total of 60 P1a proteins. The second, P1b, creates trimers at the vertices of the PC’s dodecahedral skeleton; as there are 20 vertices, these trimers also contain a total of 60 proteins.

In wild-type specimens of φ6, each P2 protein is bound to the inner surface of the procapsid at a site close to a 3-fold symmetry axis of the dodecahedral structure [98], while each P4 is assembled on the procapsid’s outer surface and overlies one
The pink, blue, red, orange, and green arrows respectively indicate the approximate positions of P1a, P1b, P2, P4, and P7 proteins in a central slice of a wild-type φ6 procapsid.

of the pentamer faces. The last protein, P7, is the least characterized protein in the PC and its precise location is still a source of debate [97, 99]. Cryo-EM studies of the related cystovirus indicate that each P7 is assembled into the PC at a site close to a 3-fold axis and the P1 shell [97]. Figure 4.8 shows the approximate positions of the proteins in a PC: The pink, blue, red, orange, and green arrows show the approximate positions of P1a, P1b, P2, P4, and P7 proteins, respectively.

To illustrate how a visualization tool based on component trees can be used to explore biological structures, we now describe an investigation of the differences between component trees of density maps of PCs of four different mutants of φ6—the wild-type version and three versions obtained by genetic deletion in which one or both of the proteins P2 and P7 is missing. The four density maps used here originate from the work of Nemecek et al. [99] and are publicly available in the EMDataBank [1]. These density maps were obtained by single particle reconstruction from electron micrographs [11].
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>EMDB access code</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC1247</td>
<td>PC wild-type</td>
<td>2341</td>
</tr>
<tr>
<td>PC124</td>
<td>PC lacking P7</td>
<td>2342</td>
</tr>
<tr>
<td>PC147</td>
<td>PC lacking P2</td>
<td>2344</td>
</tr>
<tr>
<td>PC14</td>
<td>PC lacking P2 and P7</td>
<td>2346</td>
</tr>
</tbody>
</table>

Table 4.1: The Density Maps of Four Recombinant Procapsids of Bacteriophage φ6.

The name, description and EMDB access code for the four different maps of φ6 PC.

One of the four density maps is of a PC of a wild-type φ6, which consists of the four proteins P1, P2, P4, and P7; it will accordingly be referred to as PC1247. Another of the density maps is of a PC of a genetically modified φ6 that was assembled from genes from which the sequence needed to synthesize protein P2 was deleted. This PC therefore contains just the three proteins P1, P4, and P7, and will be referred to as PC147. A third density map is of a φ6 PC that contains P1, P2, and P4 but is missing the protein P7; this PC will be referred to as PC124. The last density map is of a φ6 PC that will be referred to as PC14, because it contains P1 and P4 but contains neither of the proteins P7 and P2.

Each density map is a digital picture \((V, \pi, f)\) in which \(V\) is a \(400 \times 400 \times 400\) array of voxels. The graylevels in the original density maps (from EMDataBank) were floating point values, but we simplified the density maps by quantizing the graylevels to a set of just 256 equally spaced values represented by the integers 0, \ldots, 255, where 0 corresponds to the minimum and 255 the maximum graylevel in the original density map. Central slices and component trees of the simplified density maps are shown in Figure 4.9. Table 4.1 summarizes the density maps used in this experiment.

For the component tree of the PC14 density map (see Figure 4.9(e)), we found that \((\lambda, k)\)-simplification with parameters \(\lambda = 2\) and \(k = 800\) yielded an appropriately simplified component tree. This simplified tree is shown in Figure 4.10(a).
Figure 4.9: Central Slices and Component Trees of Mutant φ6 PCs.
Central slices of (a) PC14 (PC lacking P2 and P7); (b) PC124 (PC lacking P7); (c) PC147 (PC lacking P2); (d) PC1247 (wild-type). Component trees of (e) PC14; (f) PC124; (g) PC147; (h) PC1247.
Figure 4.10: (2,800)-Simplified Component Tree of the PC14 Component Tree. (a) The (2,800)-simplification of the PC14 component tree in Figure 4.9(e); nodes associated with a P1a pentamer, a P1b trimer, and a P4 protein are indicated by the pink, blue, and orange arrows, respectively. (b) Surface renderings of the P1a pentamer (pink), P1b trimer (blue), and P4 protein (orange) represented by the nodes indicated by the colored arrows in (a).

The node indicated by the black arrow is the parent of 12 leaves that correspond to 12 P1a pentamers, and is the grandparent of 60 leaves that correspond to 60 P1b proteins which form the 20 P1b trimers. Copies of P4, the other protein in PC14, are represented by the 12 leaves in the orange oval on the left side of the component tree. Figure 4.10(b) depicts surface renderings of the components associated with the nodes indicated by the colored arrows in (a), which represent a P1a pentamer (pink), P1b trimer (blue), and P4 protein (orange). Thus Figure 4.10 identifies the parts of the simplified component tree that represent the three substructures (i.e., the P1a pentamers, P1b trimers, and P4 proteins) which make up the PC14 procapsid.

The second component tree we investigated was the component tree of the PC124 density map. Figure 4.11(a) shows its (2,800)-simplification. Note that, with the exception of the nodes in the red region, the component tree in Figure 4.11(a) has the same structure as the (2,800)-simplified PC14 component tree in Figure 4.10(a). Again, the node indicated by the black arrow is the parent of 12 leaves that correspond to 12 P1a pentamers and 20 nodes (each with 3 leaf-children) that correspond to 20 P1b trimers. It is also a “cousin” of 12 leaves that represent
Figure 4.11: (2,800)-Simplified Component Tree for the PC124 Component Tree. (a) The (2,800)-simplification of the PC124 component tree in Figure 4.9(f). (b) Surface renderings of the nodes indicated by the pink, blue, and orange arrows in (a), which represent a P1a pentamer (pink), P1b trimer (blue), and P4 protein (orange). (c) Surface renderings of the P2 proteins represented by the two nodes indicated by the red arrows in (a).

12 P4 proteins. Copies of P2, the additional protein in PC124, are represented by the 20 leaves in the red oval on the right side of the component tree. Figure 4.11(b) shows surface renderings of the P1a pentamers, P1b trimers, and P4 proteins associated with the nodes in Figure 4.11(a) that are indicated by the pink, blue, and orange arrows respectively. The P2 proteins are on the inside of the PC shell; surface renderings of the nodes associated with two of the P2 proteins (indicated by the red arrows in Figure 4.11(a)) are shown in Figure 4.11(c).

In contrast to the simplified PC14 and PC124 component trees in Figures 4.10(a) and 4.11(a), the (2,800)-simplified PC147 component tree shown in Figure 4.12(a) has a leaf for each P1a protein rather than a leaf for each P1a pentamer: the node indicated by the black arrow in Figure 4.12(a) is still the parent of 12 nodes associated with P1a pentamers and 20 nodes associated with P1b trimers, but each of the nodes associated with the P1a pentamers now has five children—one for each P1a protein in the pentamer. The presence of a leaf for each P1a protein can be explained by a property of PC147 that is discussed in [99]: the very close proximity of each P1a to a P7. The densities from these two proteins are likely to be combined, resulting in
Figure 4.12: Identifying Proteins in PC147 Component Trees.
(a) The (2, 800)-simplification and (c) the (2, 50)-simplification of the PC147 component tree in Figure 4.9(g). (b) Surface renderings of components in the (2,800)-simplified tree in panel (a) that represent a P1a pentamer (pink), a P1b trimer (blue), and a P4 protein (orange). (d) Surface renderings of the three components indicated by the green arrows in the (2,50)-simplified tree in panel (c), which represent P7 proteins.

components that are large enough to survive step 1 of a (2,800)-simplification.

The proximity of each P1a to a P7 in PC147 also explains the absence of nodes that represent just the P7 proteins in Figure 4.12(a). But when we used a smaller parameter value $k = 50$ (instead of $k = 800$) in step 1 of the simplification process we found leaves that represent P7 proteins in the simplified component tree. In Figure 4.12(c), which shows the (2,50)-simplified PC147 tree, these leaves can be seen in the enlarged part of the tree.

Surface renderings of the nodes of the (2,800)-simplified PC147 tree that are indicated by the pink (P1a pentamer), blue (P1b trimer), and orange (P4 protein) arrows in Figure 4.12(a) are shown in Figure 4.12(b). Surface renderings of the nodes of the (2,50)-simplified PC147 tree that are indicated by the green arrows in
Figure 4.13: Identifying Proteins in PC1247 Component Trees.
(a) The \((2, 800)\)-simplification of the component tree presented in Figure 4.13(e).
(b) Surface renderings of the components associated with a P1a protein (pink), a
P1b protein (blue), and a P4 protein (orange). (c) The \((2, 300)\)-simplification of the
component tree presented in Figure 4.13(e). (d) Surface renderings of the compo-
nents indicated by the green (P7) and red (P2) arrows in (c).

Figure 4.12(c) are shown in Figure 4.12(d); as mentioned above, these components
correspond to P7 proteins.

The last density map we investigated was that of the wild-type procapsid PC1247.
Figure 4.13(a) depicts the \((2, 800)\)-simplification of the PC1247 component tree in
Figure 4.9(h). There is a node that is the parent of 60 leaves that correspond to 60
P1a proteins and the grandparent of 60 leaves that correspond to 60 P1b proteins;
there are 12 leaves, associated with P4 proteins, which are in the orange oval on the
left side of the tree.

The \((2,800)\)-simplified PC1247 component tree in Figure 4.13(a) does not show
nodes associated with the P7 and P2 proteins. However, the parameter choices
\(\lambda = 2\) and \(k = 300\) yield a \((\lambda,k)\)-simplified component tree in which nodes corre-
sponding to the P2 and the P7 proteins are easily identified. The (2,300)-simplified component tree is shown in Figure 4.13(c); the enlarged part of the tree shows the nodes associated with the P2 proteins (red shading) and the P7 proteins (green shading).

Surface renderings of the components associated with the nodes indicated by the pink (P1a pentamer), blue (P1b trimer), and orange (P4 protein) arrows in Figure 4.13(a) are shown in Figure 4.13(b). Surface renderings of the components associated with the nodes indicated by the red (P2 protein) and green (P7 protein) arrows in Figure 4.13(c) are shown in Figure 4.13(d).

These experiments show that component trees can be useful in investigating biological structures associated with microbes – in particular virus particles. We have shown through the use of four different versions of a bacteriophage procapsid that procapsid’s protein substructures correspond to particular nodes of simplified component trees. We have also seen how component trees can provide an indication that certain substructures occur in very close proximity to each other.

4.4 Case Study 2: Using Component Trees to Investigate the Structure of a Protein

Biological macromolecules are large polymers made up of linked smaller molecules. The four major types of biological macromolecules – proteins, carbohydrates, lipids and nucleic acids – perform a “combination of a diversity of functions in all living systems such as energy storage, structural support, protection, transport and regulation” [100]. The function of biological macromolecules are directly related to their shape and their chemical properties.

Proteins are macromolecules formed by various amino acids organized in specific spatial conformations. To understand the functions of proteins at a molecu-
lar level, it is often necessary to determine their three-dimensional structure [101]. Proteins can be characterized by four different hierarchical levels of complexity [100, 101, 102]. The linear order of different amino acids that compose a protein is called its primary structure. The secondary structure refers to local spatial relationship of amino acids that are close in the primary sequence. The two principal elements in the secondary structure are the $\alpha$-helix and the $\beta$-sheet.

The tertiary structure contains the complete three-dimensional conformation of a protein, describing the way that the secondary structure’s elements are arranged in space. The last level, though not less important, is the quaternary structure. This level describes the three-dimensional structure of a multi-subunit protein and how the subunits fit together. Figure 4.14 shows the four hierarchical levels that characterize proteins.

Experimental methods such as X-ray crystallography or NMR can be used to determine the secondary structure of certain proteins. However, aspects such as the size, complexity, and dynamic nature of macromolecular complexes may limit the study of some macromolecular complexes using these methods. Fortunately, the resolution determined by EM has become close to that determined by X-ray crystallography, enabling the investigation of such macromolecular complexes. This new level of resolution allows for the identification of secondary structure elements (SSE) in EM maps, aiding in the interpretation of large macromolecular complexes for which the atomic model is still unavailable.

Secondary structure elements of a protein may be identified in EM maps if the resolution distance is below 10 Å [103]. More specifically, the $\alpha$-helix can be clearly characterized at 6 Å and $\beta$-sheet can be identified at resolution distance of 5 Å. Several methodologies have been proposed to manually or automatically identify the SSE in EM density maps [103, 104, 105, 106]. A short review of proposed methodologies to identify SSE can be found in [107].
Figure 4.14: Four Protein Hierarchical Levels.
(a) Sequence of amino acids that compose a primary structure of a protein. (b) The two principal elements in the secondary structure: $\beta$-sheet (left) and $\alpha$-helix (right). (c) Tertiary structure of a protein forming a three-dimensional folding of polymers. (d) Quaternary protein structure is composed by the combination of multiples chains of polymers (Figure reproduced from [100]).
This section presents two basic examples of how the nodes of a simplified component tree can indicate the possible positions of α-helices and β-sheets in a density map. The first indication that component trees may be useful in exploring secondary structure elements occurred while investigating the major capsid protein P1 of the φ6 procapsid (PDB ID code 4K7H [108]) using MataExplorer.

In order to understand the position of P1 in a φ6 procapsid, a density map for the 4K7H atomic model was created using the program MolMap from Chimera [25]. Then, several volumes were created by the selection of nodes of a simplified component tree for the 4K7H atomic model density map using MataExplorer. Interestingly, it was found that the surface rendering for these volumes indicates positions of α-helices and β-sheets in the 4K7H atomic model.

Figures 4.15(a) and (b) show the density map at 4.89 Å (78 × 81 × 57 voxels) for the 4K7H atomic model and its simplified component tree, respectively. The surface rendering for the connected components associated with the nodes selected in the simplified component tree of 4K7H is presented in Figure 4.15(c). The colored disks in the component tree depicted in Figure 4.15(b) indicate the selected nodes used to create the surface rendering in Figure 4.15(c). Note that the surface rendering for the density map presented in Figure 4.15(a) is shown in light gray within Figure 4.15(c).

It is apparent from Figures 4.15(c) and (d) that the surface renderings of the components selected in the component tree correspond to some α-helices and/or β-sheets of the 4K7H atomic model. For example, the orange surface rendering in 4.15(c) corresponds to three α-helices and part of a β-sheet of the 4K7H atomic model. More specifically, the three α-helices indicated by the orange surface rendering in Figure 4.15(c) correspond to the two gold α-helices of the Lever subunit and one red α-helix of the Anchor subunit in Figure 4.15(d). In addition, part of a β-sheet corresponds to the yellow β-sheet in the Anchor subunit.
Figure 4.15: Using Component Trees to Explore Protein Secondary Structure.
(a) Density map at 4.89 Å obtained from the 4K7H atomic model (PDB ID code 4K7H [108]) and (b) its simplified component tree. (c) Surface rendering for the nodes selected in the component tree presented in (b). The colored disks in the component tree indicate the selected nodes used to create the surface rendering in (c). (d) The ribbon diagram for the 4K7H atomic model. The colors in Figures (c) and (d) are not related in any way. (Figure (d) is reproduced from [108]).
The two examples presented in this section used component trees obtained from data simulated from atomic models downloaded from the wwPDB [3]. They are based on the experiments reported by [104, 106], where protein atomic models representing classes of the Structural Classification of Proteins (SCOP) [109] were used. The SCOP classification organizes proteins structures into hierarchical levels based on their evolutionary and structural relationship. SCOP is one of the most widely accepted protein structure classification systems used in the study of biological structures.

Example 1

In this example, an atomic model of the bacteriorhodopsin (PDB ID code 1C3W [110]) is used. The bacteriorhodopsin structure contains 7 α-helices and one β-sheet and was selected in order to represent the α-helix class according to SCOP. Figure 4.16(a) shows the ribbon diagram for atomic model 1C3W. The α-helices are plotted in yellow, while the β-sheet is plotted in light blue.

In order to compute the component tree for this and the other experiment reported in this section, density maps at 8 Å were created from PDB atomic models using the program MolMap from the software package Chimera [25]. Figure 4.16(b) shows the surface rendering for the density map created from the 1C3W atomic model. For this first example, a simplified component tree was computed using the parameters $\lambda = 0$ and $k = 10$. Figure 4.16(c) shows the simplified component tree for the IC3W density map.

Using MataExplor, each component associated with a leaf of the component tree presented in Figure 4.16(c) was saved as a new density map and then plotted using Chimera [25]. Figure 4.16(d) shows the surface rendering for the seven components associated with the leaves in the tree presented in Figure 4.16(c). The colored disks in the component tree presented in Figure 4.16(c) indicate the selected nodes used
Figure 4.16: Secondary Structure Elements for the Bacteriorhodopsin.
(a) The ribbon diagram for the bacteriorhodopsin atomic model (PDB ID code 1C3W [110]). (b) Density map at 8 Å and (c) (0, 10)-simplified component tree of the density map for the bacteriorhodopsin. (d) Surface rendering for the components associated with the leaves of the component tree presented in (c). The colored disks in the component tree indicate the selected nodes used to create the surface rendering in (d). To show the $\alpha$-helix associated with the leaves of the component tree in Figure 4.16(c), the surface rendering of these leaves is fitted in the ribbon diagram in (e).
to create the surface rendering in Figure 4.16(d).

A close examination of the leaves of the component tree presented in Figure 4.16(c) reveals an interesting fact: the surface rendering for the components associated with these leaves has a spatial organization very similar to the seven \( \alpha \)-helix elements of the atomic model 1C3W presented in Figure 4.16(a). Note that each of the 6 green surface renderings depicted in Figure 4.16(d) indicate the position of one \( \alpha \)-helix, while the cyan surface indicates a position of one \( \alpha \)-helix and one \( \beta \)-sheet. The proximity of the \( \alpha \)-helix and the \( \beta \)-sheet in the secondary structure could be the reason that these two elements belong to the same component in Figure 4.16(d). Figure 4.16(e) shows the ribbon diagram for the 1C3W atomic model fitted with the surface rendering of the components that are shown in Figure 4.16(d).

Example 2

For the second example, a density map obtained from a subunit of the atomic model for triose phosphate isomerase (PDB ID code 1TIM [111]) was used. This subunit contains 10 \( \alpha \)-helices on the outside and 10 \( \beta \)-sheets on the inside. The model was selected in order to represent the \( \alpha \)-helix/\( \beta \)-sheet class according to the SCOP. Figures 4.17(a) and 4.17(b) show the ribbon diagram and the density map at 8 Å for a subunit of the 1TIM atomic model. The numbers in Figure 4.17(a) indicate the position of the 10 \( \alpha \)-helices in the 1TIM atomic model.

The simplified component tree was computed using the parameters \( \lambda = 0 \) and \( k = 15 \). Figure 4.17(c) shows the simplified component tree for the 1TIM density map. Again, MataExplor was used to create a volume for each leaf in the component tree presented in Figure 4.17(c). The surface rendering for these components is shown in Figure 4.17(d). The colored disks in the component tree presented in Figure 4.17(c) indicate the selected nodes used to create the surface rendering in 4.17(d).
Note that the eight green surface renderings depicted in Figure 4.17(d) correspond to the eight $\alpha$-helices of Figure 4.17(a), while the cyan surface rendering corresponds to one $\alpha$-helix and two $\beta$-sheets. The two red surface renderings in Figure 4.17(d) correspond to different parts of the $\beta$-sheets in the 1TIM atomic model. Figure 4.17(e) shows the ribbon diagram for the 1TIM atomic model fitted with the surface rendering of the components that are shown in Figure 4.17(d).

The only $\alpha$-helix of the 1TIM atomic model that was not identified by the component tree presented in Figure 4.17(c) was the $\alpha$-helix numbered 9 in Figure 4.17(a). In Figure 4.17(e), the missed $\alpha$-helix is depicted in pink.

These two examples presented in this section provide some evidence to support the claim that component trees of density maps computed from atomic models may be useful for identifying the position of secondary structure elements in macromolecular complexes. However, further investigation is needed to confirm this. Nevertheless, these examples illustrate how component trees can produce useful segmentations of density maps that aid in the study of macromolecules.
Figure 4.17: Secondary Structure Elements for the Triose Phosphate Isomerase. (a) Ribbon diagram for the atomic model and (b) density map at 8 Å for the bacteriorhodopsin (PDB ID code 1TIM [111]). (c) The $(0, 10)$-simplified component tree of the density map presented in (b). (d) The surface rendering for the components associated with the leaves of the component tree presented in (c). The colored disks in the component tree indicate the selected nodes used to create the surface rendering in (d). (e) The ribbon model fitted to the surface rendering of the components associated with the leaves.
Chapter 5

Investigation of the Use of Component Trees for Macromolecular Docking

One of the core tasks to be accomplished in the docking procedure is to find the spatial relationship between a complete biological specimen and one or more of its subunits. One of the aims of this study is to investigate a graphic-algorithmic approach based on component trees to reduce the human labor (work-intensity) and computer burden of docking a high-resolution subunit into a low-resolution macromolecular complex. More specifically, this research aims to understand how component trees can guide the docking process by producing a list of approximate positions (or even the exact positions) of the subunits in the complete macromolecular assembly.

Our initial work on component trees and on embedded digital spaces suggests that component trees could be used to study biological structures and to explore the relationship between objects at different resolutions. This chapter presents a preliminary investigation of the use of methodology that utilizes component trees to aid in the process of docking of a high-resolution subunit into a low-resolution
5.1 A Tentative Docking Methodology Based on Component Trees

Oftentimes, the same object is imaged in different ways, resulting in digital pictures of (some parts of) it at different resolutions. In Section 3.1, the first steps towards developing a theory for the intuitive idea of embedded digital spaces and digital pictures were presented. The concept of embedded digital space was introduced to formalize the relationship between digital pictures of the same object at different resolutions.

The idea behind this definition is that a group of elements in the high-resolution digital picture are in some way combined to create a single element in the low-resolution digital picture. In our definitions we assume that if a digital space \((V', \pi')\) is embedded in the digital space \((V, \pi)\), the set \(V'\) is a partition of \(V\) and \(\pi'\) defines connected sets in the elements of \(V'\) based on the \(\pi\)-connectedness of \(V\).

Usually macromolecular docking techniques use two digital pictures: a target and a model. These techniques search for a “region” in the target digital picture that describes the object represented by the model digital picture. In the macromolecular docking context, the target digital picture is usually a low-resolution density map that describes the whole macromolecular complex. On the other hand, the model digital picture is usually a high-resolution density map that represents a subunit of a macromolecular complex. The task for macromolecular docking is to identify the previously mentioned region in the target digital picture and then fit the the model digital picture into this region.

The basic idea of applying tree algorithms to macromolecular docking is to use component trees to perform the search for that desired region in the target digital
picture. More specifically, tree algorithms can be used to explore the target component tree (the component tree for the target digital picture) in order to find a subtree that has approximately the same structure that the model component tree (the component tree for the model digital picture). This idea is based on the ability of a component tree to capture the essential structural information of a biological specimen independent of the resolution or the device used to image the macromolecule. Remember that component trees manifest the relationship between the parts of a biological structure, irrespective of the physical process (such as cryo-EM or X-ray crystallography) that was used to obtain the density map.

Suppose for example, that one desires to know if a high-resolution density map for a specific subunit belongs to a whole macromolecular complex. The answer to this question can be found by analyzing the component trees for the subunit and the whole macromolecular complex. Let $T$ be a component tree for a macromolecular complex and $T'$ be a component tree for one of it subunit. If the subunit is present in the macromolecular complex, it is likely that there exists a subtree of $T$ that has approximately the same structure as $T'$. The challenge here is to identify one (or a list of) subtree(s) among all the subtrees of the component tree for the whole macromolecular complex that better represents the subunit.

Based on the initial investigation of the relationship of digital pictures at different resolutions presented in Section 3.1, the use of tree embedding is likely to be suitable in the search for a subtree of the target component tree that represents the structural information of the model density map. Tree embedding preserves the hierarchical relationship between the nodes in the component trees allowing flexibility of what node will be used in the map. In other words, a tree embedding is a mapping of the nodes of one rooted tree into the nodes of another rooted tree that preserves the ancestor-descendant relationships among nodes. The concept is otherwise totally flexible (in the sense that its definition imposes no other constraints)
Figure 5.1: A Tree Docking Methodology Pipeline.

Substep (i) - compute the component tree for the two density maps; substep (ii) - identify the embedding maps that maximize a measure of similarity; substep (iii) - use a selected embedding map $\varphi_e$ to fit the high-resolution density map into the low-resolution density map.

The docking methodology proposed in this chapter can be understood as consisting of three substeps in the STEP 3 docking strategy presented in Section 2.2. In substep (i) (of STEP 3), we compute and simplify component trees of the target and the model digital pictures. In substep (ii), we search within the target’s simplified component tree for a subtree that has approximately the same structure as the model’s simplified component tree. In substep (iii), we attempt to fit the model digital picture into the region of the target digital picture that is represented by the subtree found in substep (ii). Figure 5.1 presents the pipeline for the proposed docking methodology.

As described in Section 3.2, the computation and simplification of component trees mentioned in substep (i) can be done using the intensities in a digital picture. The core of this methodology is to define a measure of similarity to identify tree embeddings between subtrees of the target component tree into the model component tree. In other words, the proposed methodology must identify a list of possible tree embeddings and use these maps to indicate the possible docking locations in
substep (iii). An initial idea for identifying such tree embeddings is described in the next few paragraphs.

Component trees, as we have defined them in this dissertation, are labeled trees in which the label of a node represents its level. However, this is not an appropriate manner of labeling a component tree for docking purposes; the target and model density maps are likely to have been obtained by very different imaging methodologies and so the physical meaning of a node’s level is unlikely to be similar in the two cases. The idea is to provide a reasonable measure of the goodness of a tree embedding \( \varphi \) (for docking purposes) by the similarity between the labels assigned to the nodes in a candidate subtree of the target’s component tree and the labels assigned to their respective image nodes under \( \varphi \) in the model’s component tree.

Recall that a **tree embedding** of a rooted tree \( T' = (N', E') \) into a rooted tree \( T = (N, E) \) is a mapping \( \varphi_e : N' \rightarrow N \) such that \( C' \) is a descendant of \( D' \) in \( T' \) if, and only if, \( \varphi_e(C') \) is a descendant of \( \varphi_e(D') \) in \( T \). The tree embedding \( \varphi_e \) is said to be **root-to-root** if \( \varphi_e(\text{root}(T')) = \text{root}(T) \).

A **labeled tree** is a triple \( (N, E, \Omega) \), where \( (N, E) \) is a rooted tree and \( \Omega : N \rightarrow \mathbb{R}_+ \). Let \( T = (N, E, \Omega) \) and \( T' = (N', E', \Omega') \) be two labeled trees. We say that \( \varphi_e \) is a **labeled tree embedding** (respectively, a **root-to-root labeled tree embedding**) of \( T \) into \( T' \) if \( \varphi_e \) is a tree embedding (respectively, a root-to-root tree embedding) of \( (N, E) \) into \( (N', E') \).

Let \( T = (N, E, \Omega) \) and \( T' = (N', E', \Omega') \) be two labeled trees and \( \varphi_e \) be a labeled tree embedding of \( T' \) into \( T \). We define the **component inconsistency** of \( \varphi_e \) as

\[
\sum_{D' \in N'} \left| \Omega'(D') - \Omega(\varphi_e(D')) \right|.
\]  

(5.1)

We use \( \omega_{T', T} \) to denote the minimal component inconsistency over all root-to-root tree embeddings of \( T' \) into \( T \).
Let $T = (N, E, \Omega)$ be any labeled tree and $C$ be an element of $N$. We call the labeled tree $T_[C] = (N_C, E_C, \Omega_C)$ the labeled subtree of $T$ at $C$ if

- $N_C$ is the set of all descendants of $C$ in $T$;
- $E_C = E \cap (N_C \times N_C)$;
- $\Omega_C$ is the restriction of $\Omega$ to $N_C$.

For any two labeled trees $T = (N, E, \Omega)$ and $T' = (N', E', \Omega')$, let $\sigma_{T', T}$ denote the minimal value of $\omega_{T'[C], T}$ over all $C' \in N'$. We aspire to discovering an algorithm that, for a given pair of labeled trees $T = (N, E, \Omega)$ and $T' = (N', E', \Omega')$, finds all combinations of nodes $C' \in N'$ and root-to-root labeled tree embeddings $\tilde{\phi}$ of $T'[C']$ into $T$ such that the component inconsistency of $\tilde{\phi}$ is $\sigma_{T', T}$.

Using the notation we have just defined, our tentative proposal for carrying out substep (ii) of our docking methodology can be restated as follows: Let $T_{\text{model}} = (N, E, \Omega)$ and $T_{\text{target}} = (N', E', \Omega')$ be labeled component trees of the model and the target digital pictures, respectively. Then the tentative proposal is to find a node $C' \in N'$ and a root-to-root labeled tree embedding $\varphi$ of $T'_{[C']}$ into $T_{\text{model}}$ for which the component inconsistency of $\varphi$ is $\sigma_{T_{\text{target}}, T_{\text{model}}}$. We are hopeful that, if we use appropriate node labelings in $T_{\text{model}}$ and $T_{\text{target}}$, then such a node $C'$ and embedding $\varphi$ will often provide a good indication of where and how to dock the model into the target, and so allow us to carry out substep (iii).

As a concrete example of how this might be done, suppose our high-resolution model is a density map of a protein generated from the X-ray coordinates found in the PDB and our low-resolution target is a density map of a macromolecular complex which contains that protein. Let $T_{\text{model}}$ and $T_{\text{target}}$ be labeled component trees of these images, and suppose we have succeeded in finding a node $C'$ of $T_{\text{target}}$ and an embedding $\varphi$ of $T'_{[C']}$ into $T_{\text{model}}$ that have the above-mentioned properties. We would then consider placements of the protein that put its center of mass near
the center of mass of the region (in the target) that comprises all voxels of the component $C'$. (We might also look for positions and orientations of the protein such that the centers of mass of the nodes $D'$ of the subtree $T_{[C']}^{\text{target}}$ are not far from the centers of mass of the nodes $\varphi(D')$ of the protein. But if more than one root-to-root embedding $\varphi$ of $T_{[C']}^{\text{target}}$ into $T_{\text{model}}$ has a component inconsistency that is fairly close to $\sigma_{T_{\text{target}}, T_{\text{model}}}$, then we should do this for all such embeddings $\varphi$ if we do it for one of them.)

This proposed methodology depends on our being able to find an efficacious way to label the nodes of the component trees of the target and the model. In the example we discuss below we label each node with its component volume (as defined by (4.1)), but this labeling may not give satisfactory results in other docking problems. It is also entirely possible that better results would be obtained if instead of the component inconsistency (5.1) we used another measure of the badness of a root-to-root labeled tree embedding. For instance, we might consider replacing $N'$ with $N' - \{\text{root}(T')\}$ in (5.1). These are issues that may be address in a extension of this research.

## 5.2 A Simple Example of Docking Using Component Trees

In this section, an illustrative example that uses the proposed docking methodology is given. Although our ultimate goal is to propose a method for docking three-dimensional model digital pictures into three-dimensional target digital pictures, in the simple example we present here, the model and the target digital pictures are two-dimensional. In fact, they are derived from slices of two similar macromolecular complexes at different resolutions.

The first macromolecular complex, a native GroEL (EMDB access code 5001
Figure 5.2: An Illustrative Example of the Proposed Docking Methodology. Surface renderings of GroEL at 4.2 Å resolution distance [112] and GroEL + GroES at 7.7 Å resolution distance [113] are shown in (a) and (d), respectively. In (d), the top ring is the GroES ring and both the middle and the bottom rings are GroEL rings. (b) shows a slice from the middle of the bottom ring of the GroEL density map (left), and a cropped region of this slice (right); the green curve encloses the region that was cropped. We use the cropped region as our model image. (e) shows a slice from the middle of the lower GroEL ring of the GroEL + GroES density map; we use this slice as our target image. Component trees of the model image (the cropped region of the slice from the GroEL density map) and of the target (the GroEL slice from the GroEL + GroES density map) are shown in (c) and (f), respectively. The surface renderings were produced using Chimera [25] and the slices were selected using XMIPP [38, 39].

[112]; the claimed resolution distance is 4.2 Å), is composed of 14 identical copies of the same chaperonin protein that are organized in two circular rings of seven proteins each. Its 200 × 200 × 200 digital picture has a voxel spacing of 1.06 Å. The second macromolecule is a GroEL + GroES in the ATP-bound state (EMDB access code 1180 [113]; the claimed resolution distance is 7.7 Å). This macromolecular complex is composed of 21 chaperonin proteins. In addition to a GroEL double-ring of 14 chaperonin proteins, this macromolecular complex has an extra circular GroES ring that consists of 7 chaperonin proteins. Its 192 × 192 × 192 density map
has a voxel spacing of 1.40 Å. Figures 5.2(a) and (d) show surface renderings of these two density maps. In Figure 5.2(d) the GroES ring is at the top; the other two rings are GroEL rings.

One slice was extracted from each of the two density maps, and the densities in these two slices were then quantized to a set of 20 equally spaced values. The slices that were extracted are from (approximately) the middle of the bottom GroEL ring in Figures 5.2(a) and (d).

The slice from GroEL + GroES at 7.7 Å resolution distance was used as the target image. This image is shown in Figure 5.2(e). To create the model image (i.e., the image that needs to be fitted into the target image) from the slice of GroEL at 4.2 Å resolution distance, we manually selected a part that corresponds to one of the 7 proteins that the slice passes through. The slice of GroEL and the model image extracted from it are shown in Figure 5.2(b); the model image is the region of the slice that is enclosed by the green contour.

As each of the two slices was extracted from the middle of a ring of GroEL chaperonin proteins, the labeled component tree of the target image should have parts that have approximately the same structure as the labeled component tree of the model image. As discussed in the previous section, substep (i) of the our tentative docking methodology is to construct simplified component trees of the model and the target digital pictures. Un simplified component trees of the model and the target images are shown in Figures 5.2(c) and (f), respectively. These trees were simplified using the methodology described in Subsection 3.2.2 to eliminate small components produced by noise or by the cropping process used to create the model image. The (1, 10)-simplification of the target component tree and the (2, 10)-simplification of the model component tree are shown in Figures 5.3(a) and 5.3(b), respectively. Recall that with the possible exception of the root, all the nodes of a simplified component tree are critical. So every node in Figures 5.3(a) and (b)
that is not a leaf and also is not the root of its tree is represented by a horizontal segment.

Importantly, even though the simplified component trees in Figures 5.3(a) and 5.3(b) have a much simpler structure than their unsimplified versions, they capture the essential structural information in the images. The tree shown in Figure 5.3(a), for example, has a subtree for each of the seven chaperonin proteins that appear in the target image. The roots of these subtrees are the nodes with exactly two or three leaf children. Compared with the unsimplified tree in Figure 5.2(f), the tree of Figure 5.3(a) is simpler mainly due to the pruning away of small components (see Subsection 3.2.2); the choice \( k = 10 \) results in the removal of approximately 40% of the nodes.

To carry out substep (ii) of Figure 5.1, we first label every node in the simplified component trees with the component volume of the connected component. For example, if a node in the component tree of the target image contains 20 voxels, then the label of that node will be \( 20 \times 1.40 \times 1.40 = 54.88 \text{ Å}^3 \) (as 1.40 Å is the voxel spacing for the target image). A node in the labeled component tree of the model image with the same number of voxels would have the label \( 20 \times 1.06 \times 1.06 = 23.82 \text{ Å}^3 \).

Let \( T = (N, E, \Omega) \) and \( T' = (N', E', \Omega') \) be the labeled component trees of the model and the target images, respectively. To complete substep (ii), we find a node \( C' \in N' \) and a root-to-root labeled tree embedding \( \varphi_e \) of \( T'_{[C']} \) into \( T \) such that the component inconsistency of \( \varphi_e \) is \( \sigma_{T',T} \).

It turns out that the green node in Figure 5.3(a) is a node \( C' \in N' \) for which such an optimal embedding \( \varphi_e \) exists. So the region of the target image that is given by the green node in Figure 5.3(a) is a region where our methodology suggests that the model be fitted. This is shown as the colored region in Figure 5.3(c).

Note that this green region is in fact the position of one of the proteins in the
Figure 5.3: Using Component Trees for Macromolecular Docking.
(a) $(1, 10)$-simplified component tree constructed from the target digital picture, and a subtree whose nodes constitute the domain of an optimal root-to-root tree embedding into the simplified component tree of the model digital picture (see (b)). The nodes of the subtree are labeled with their component volumes (measured in cubic angstroms). (b) $(2, 10)$-simplified component tree constructed from the model image; each node is labeled with its component volume. (c) The target image; the green region in (c) is the region that is represented by the subtree shown in (a): This is one possible location where our methodology suggests that the model digital picture, shown in (d), should be fitted.
5.3 A Proposed Evaluation Methodology

A literature review of macromolecular docking reveals several *Figures of Merit* (FOMs) that are used to verify the accuracy of a given fitting methodology. The task to be evaluated is the finding of the position of the model density map (or the atomic structure associated with the model density map) in the target density map. The basic assumption for these FOMs is that the correct position of the model density map in the target density map is known. Here we propose the use of two FOMs: *Root Mean Square Deviation* (RMSD) and *Orientation Score* (OS). RMSD is extensively used as a measure of accuracy in macromolecular docking [6, 30, 47, 114]. The second measure, OS, was used by Pintilie *et al.* [114] and is based on the amount of effort needed to align the model density map from the estimated docking position to the correct docking position by rigid-body shift and rotation.

RMSD is used to measure the average distance between the correct docking position and the estimated docking position (i.e., the position for the model density map computed by a docking methodology). RMSD is computed from two different types of information: the coordinates for all the voxels in the high-resolution density map (RMSD$_j$) and the coordinates for all the atoms in the atomic model associated with the high-resolution density map (RMSD$_h$).

Let $(V, \pi, f)$ be the high-resolution digital picture (i.e., the model density map) for the atomic model that needs to be fitted in a low-resolution digital picture $(V', \pi', f')$ (i.e., the target density map). The function $j_{\mathcal{R}}$ is defined to give the correct docking position for an element of $V$ in $(V', \pi', f')$. Similarly, we define a function $j^*_{\mathcal{R}}$ that gives the estimated docking position for an element of $V$ in $(V', \pi', f')$. 

GroEL + GroES slice. So in this example, the docking suggestion given by our tentative methodology is correct.
Again, the estimated position is computed by the present docking methodology. Using these two functions, \( j^*_\mathcal{A} \) and \( j\mathcal{A} \), we can compute RMSD\(_j\) as follows:

\[
\text{RMSD}_j = \sum_{c \in V} (|j\mathcal{A}(c) - j^*_\mathcal{A}(c)|),
\]

in which the individual differences are called residuals.

To compute RMSD\(_h\), instead of using all the positions for the elements of \( V \), only the positions of the atoms in the atomic model are used. Let \( \mathcal{A} \) be the set of all atoms in the atomic model that need to be fitted in \((V', \pi', f')\). We define the functions \( h\mathcal{A} \) and \( h^*_\mathcal{A} \) to be the functions that give the correct and estimated docking position for an atom in the fitted atomic model. Using the functions \( h\mathcal{A} \) and \( h^*_\mathcal{A} \) we compute RMSD\(_h\) as:

\[
\text{RMSD}_h = \sum_{a \in \mathcal{A}} (|h\mathcal{A}(a) - h^*_\mathcal{A}(a)|).
\]

RMSD\(_j\) and RMSD\(_h\) are extensively used as a measure of accuracy in macromolecular docking [6, 30, 47, 114]. A small value of RMSD\(_j\) and RMSD\(_h\) indicates an accurate fit. Figures 5.4(a) and (b) present an illustration of how RMSD\(_j\) and RMSD\(_h\) can be computed.

The second measure used as a FOM is the Orientation Score (OS). As mentioned previously, this measure is based on the amount of effort necessary to align the model density map from the estimated docking position to the correct docking position by rigid-body shift and rotation. The first component of OS, \( \mathcal{D} \), is the distance between the centers of the mass of the model map in the estimated and in the correct docking positions. If \( \mathbf{x}_c \) and \( \mathbf{x}_e \) are the centers of the mass of the estimated and the correct docking positions respectively, then

\[
\mathcal{D} = |\mathbf{x}_c - \mathbf{x}_e|.
\]
Figure 5.4: Figures of Merit: RMSD and OS.

The rectangles with the black border indicate the correct docking position and the rectangles with the blue border indicate the estimated docking position. In (a), the red line shows the distance between an element of $V$ in the correct and in the estimated docking position, while in (b) it indicates the distance between an atom in the set $\mathcal{A}$ in the correct and in the estimated docking position. The green circles in (b) represent the elements of the set $\mathcal{A}$. The red line in (c) shows the distance between the centers of mass of the digital picture in the correct and in the estimated docking positions. In (d), the red line shows the rotation necessary to align the digital picture from the estimated to the correct docking position after the alignment of the centers of mass.

The second component, $ang$, is the angle rotation necessary to align the model density map from the estimated to the correct docking position after the alignment of the centers of mass. Let $(V, \pi, f)$ be the high-resolution digital picture for the atomic model that needs to be fitted in a low-resolution digital picture $(V', \pi', f')$. Let $x'_c$ and $x'_e$ be the vector positions of a voxel $c \in V$ for the correct and estimated docking positions respectively, with respect to the (common) center of mass. It should be the case that, for all $c \in V$,

$$x'_e = x'_c M',$$

(5.5)
where \( M' \) is a rotation matrix. The angle rotation \( \text{ang} \) is the one defined by this rotation matrix \( M' \).

Examples of using OS as a FOM for macromolecular docking can be found in [6, 114]. A small value of \( D \) and \( \text{ang} \) indicate an accurate fit. Figures 5.4(c) and (d) show how the orientation score can be used to measure the docking accuracy.

In what follows we suggest two types of experiments for evaluating the efficacy of docking methods. We have not yet implemented these suggestions; doing so is part of our proposed future research.

For the first type of experiment, synthetic data sets are used to investigate the relationship between digital pictures at different resolutions, their component trees, and how these trees can be employed to accomplish docking. A data set for such an experiment can be produced using the following procedure: (i) create a high-resolution digital picture \((V, \pi, f)\) that are composed of several geometric objects with elements of component volume \( \omega \); (2) for a \( \omega < \omega' \), create a low-resolution digital picture \((V', \pi', f')\) that is embedded in \((V, \pi, f)\) with \( \omega' \) as the component volume of the elements of \( V' \) (recall that, by the definition of embedding, \( V' \) is a partition of \( V \)); (iii) for a \( V_p \subseteq V \), create a high-resolution digital picture \((V_p, \pi, f)\). This procedure creates three digital pictures: \((V, \pi, f)\), \((V_p, \pi, f)\), and \((V', \pi', f')\).

Because the shapes and positions of the geometrical objects in the digital pictures \((V, \pi, f)\), \((V_p, \pi, f)\) and \((V', \pi', f')\) are known, a careful investigation can be done to verify the docking accuracy and to see the effect of the \((\lambda, k)\)-simplification methodology in the docking process. Figure 5.5 illustrates the idea of using synthetic digital pictures for this first type of proposed experiment.

For the second type of experiment, the data set proposed in the EM Modeling Challenge [115] is to be used. This challenge was proposed by The National Center for Macromolecular Imaging in 2010 with the aim of opening a communication channel between the modeling and the cryo-EM community. This initiative
Figure 5.5: Demonstration of the Proposed Experiments with Synthetic Data. The digital picture \((V', \pi', f')\) is regularly embedded in the digital space \((V, \pi, f)\). Each element of \(V'\) is a \(4 \times 4\) subarray of the pixels in \(V\). The left blue ellipse indicates a region \(V_p\) of \(7 \times 7\) pixels, selected to create the digital picture \((V_p, \pi, f)\). The right blue ellipse indicates the region where \((V_p, \pi, f)\) should be fitted in \((V', \pi', f')\).

provides resources in which the community can test their methodology, exchange ideas, and compare their results.

The EM Modeling Challenge database is composed of 13 Cryo-EM density maps in which the resolution distance range spans from 2.5 Å to 23.5 Å. This database includes examples of large and highly symmetric viruses, smaller and intermediate symmetry assemblies, and completely asymmetric structures. With this diversity of resolution distance and symmetry, the intention of the EM Modeling Challenge is to provide a database that is suitable for map segmentation, protein-backbone tracing, as well as both rigid and flexible docking.

One of the advantages of using this data set is the possibility of comparing the results obtained by the proposed methodology with other methodologies that used the same database. At this point more than 20 different methodologies have used this database for rigid-body and flexible macromolecular docking.

Each data set in the EM Modeling Challenge database is composed by a target density map and an atomic model for which the position in the target density maps is known a priori. A criterion for selecting data sets is to choose target density
maps in the database that describe similar macromolecules at different resolutions
distance. For example, the density maps selected could be GroEL at 4.2 Å resolution
distance, GroEL + GroES at 7.7 Å resolution distance, and GroEL + GroES at
23.5 Å resolution distance.

GroEL is a large oligomer composed of 14 subunits that span two circular rings
of seven proteins each. Each of these subunits are identical copies of the same
protein (molecule of a chaperonin of about 60 kDa) that is represented by 14 chains
labeled $A, B, \ldots, N$. The GroEL + GroES complex consists of three rings (including
the two rings from the GroEL) with chains labeled from $A, B, \ldots, U$. For the GroEL
at 4.2 Å resolution distance, chain $A$ from the atomic model with PDB ID 1XCK
[116] can be selected to be fitted, while chain $A$, $H$, and $O$ (one from each ring)
from the atomic model with PDB ID 2C7C [113] can be selected to be fitted into
the GroEL + GroES at resolution distance 7.7 Å and 23.5 Å. Figure 5.6 shows the
surfaces for these three density maps and the atomic model.

In order to perform such experiments, a model digital picture for the PDB file
for the selected chains has to be created. To isolate a chain from an atomic model,
we can use tools such as Chimera or simply edit the PDB file manually. Then, a
digital picture for the isolated chain is created using the methodology proposed by
[13] (the Low-pass Electron Atomic Scattering Factors - LEASF) that is currently
available through the XMIPP software [38, 39].

The results finding from the second type of experiments can then be compared
with other results reported in the EM Modeling Challenge for the same target and
model digital pictures; for example, by Pintilie and Chiu [114] and Tijoe et al.
[118].
Figure 5.6: Experimental Data sets.
The solid surface for (a) GroEL at 4.2 Å resolution distance (EMDB access code 5001 [112]), (b) GroEL + GroES at 7.7 Å resolution distance (EMDB access code 1180 [113]) and (c) GroEL + GroES at 23.5 Å resolution distance (EMDB access code 1046 [117]). The two atomic models that can be used in the proposed experiments are PDB ID (d) 1XCK [116] and (e) 2C7C [113]. In the atomic models, the green indicate the chain A, blue the chain H and purple the chain O. Figures (a)–(e) were produced using Chimera [25].
Chapter 6

Conclusion

6.1 Contributions

The following are the main contributions of this dissertation:

- A novel interactive framework for visual exploration of component trees of density maps of macromolecular complexes, with the purpose of improved understanding of their structure.

- Proposed the \((\lambda,k)\)-simplification methodology and proved its robustness in the presence of noise.

- Investigated a mathematical theory for digital pictures that represent objects at different resolutions.

- Conducted an experimental investigation of the use component trees and their simplifications as aids in the exploration of macromolecular structures. These experimental studies included:
  
  – identification of the difference between two very similar viruses,
– showing how differences between component trees reflect the fact that structures of mutant virus particles have varying sets of constituent proteins,

– utilizing component trees for density map segmentation in order to identify substructures within a macromolecular complex,

– showing how an appropriate component tree simplification may reveal the secondary structure in a protein, and

– outlining a potential strategy for docking a high-resolution representation of a substructure into a low-resolution representation of whole structure.

In the course of writing this dissertation, the topics listed above were published in various scientific media such as peer-review journals, books, and conferences. The list of publications associated with this dissertation are:


6.2 Suggestion for Future Work

The research presented in this dissertation has demonstrated the potential usefulness of component trees for the understanding of the three-dimensional structure of macromolecular complexes. Here we list some directions for further investigation of the use of component trees for exploring digital pictures.

• Further investigate the use of component trees for macromolecular docking;

• Investigate a heuristic for automatically finding appropriate \( \lambda \) and \( k \) parameters in the \((\lambda,k)\)-simplifications;

• Further study the use of component trees for secondary structure identification in macromolecules;

• Examine the use of component trees and \((\lambda,k)\)-simplifications for different modalities of images such as MRI, X-ray, and CT.
Appendix A

Some Properties of Simplification

Steps 2 and 3, and a Proof of the Correctness of Algorithm 1

A.1 Properties of Simplification Step 2

We prove the main result of Sub-subsection 3.2.4.2 and establish other properties of the \((\lambda, k)\)-simplification step 2 that are used in our proof of Theorem 4.

**Lemma A.1.** Let \(F_{in} = (T_{in}, \ell_{in})\) be any FCTS, let \(\lambda > 0\), and let \(S\) and \(S'\) be any two distinct leaves of an FCTS \(F_{out} = (T_{out}, \ell_{out})\) that results from pruning \(F_{in}\) by removing branches of length \(\leq \lambda\). Then (regardless of which \(\ell_{in}\)-increasing enumeration of \textbf{Leaves}(\(T_{in}\)) is used to perform the pruning):

\[
\begin{align*}
(i) & \quad \land_{T_{out}} \{S, S'\} = \land_{T_{in}} \{S, S'\} \\
(ii) & \quad \min(\ell_{out}(S), \ell_{out}(S')) - \ell_{out}(\land_{T_{out}} \{S, S'\}) > \lambda
\end{align*}
\]

---

\(^1\)Appendices A, B and C are parts of a book chapter published in [20]. The main contributor to the proof presented here is Dr. Yung Kong from the Computer Science Department, Queens College, City University of New York (ykong@cs.qc.cuny.edu).
Proof. The hypotheses imply that properties P1–P4 of Subsection 3.2.4 hold with respect to some $\ell_{in}$-increasing enumeration of $\text{Leaves}(T_{in})$. It follows from P4 that, for all $A \in \text{Nodes}(T_{out})$, every node in $A \upharpoonright T_{in}$ is also a node in $A \upharpoonright T_{out}$. Therefore $A \upharpoonright T$ is the same set regardless of whether $T = T_{out}$ or $T = T_{in}$. So $\Lambda_T \{S, S'\}$ is the same node regardless of whether $T = T_{out}$ or $T = T_{in}$, since $\Lambda_T \{S, S'\}$ is just the element of $S \upharpoonright T \cap S' \upharpoonright T$ that is a descendant in $T$ of every element of that set. Hence (i) holds.

To prove (ii), we may assume without loss of generality that, in the $\ell_{in}$-increasing leaf enumeration that is used for pruning, $S$ occurs later than $S'$. (This assumption implies that $\min(\ell_{in}(S), \ell_{in}(S')) = \ell_{in}(S')$.) Then, since $S' \in \text{Leaves}(T_{out})$, property P3 implies that $\ell_{in}(S') - \ell_{in}(\Lambda_{in} \{S, S'\}) > \lambda$, which is equivalent to:

$$\min(\ell_{in}(S), \ell_{in}(S')) - \ell_{in}(\Lambda_{in} \{S, S'\}) > \lambda \quad (A.1)$$

But Equation (A.1) is equivalent to assertion (ii), because of assertion (i) and the fact that $\ell_{out}$ is just the restriction of $\ell_{in}$ to $\text{Nodes}(T_{out})$. \hfill $\square$

Corollary A.1. Let $\lambda$ be any positive value, and $\mathfrak{F}_{out}$ any FCTS that results from pruning an FCTS $\mathfrak{F}_{in}$ by removing branches of length $\leq \lambda$. Then, for all $A \in \text{Crit}(\mathfrak{F}_{out}) \setminus \text{Leaves}(\mathfrak{F}_{out})$, we have that $A \in \text{Crit}(\mathfrak{F}_{in}) \setminus \text{Leaves}(\mathfrak{F}_{in})$ and depth$_{\mathfrak{F}_{out}}(A) > \lambda$.

Proof. Let $\mathfrak{F}_{out} = (T_{out}, \ell_{out})$, and let $A \in \text{Crit}(\mathfrak{F}_{out}) \setminus \text{Leaves}(\mathfrak{F}_{out})$. Then $A = \Lambda_{out} \{S, S'\}$ for some distinct leaves $S$ and $S'$ of $\mathfrak{F}_{out}$. Now $A = \Lambda_{in} \{S, S'\}$ (by assertion (i) of Lemma A.1), and so $A \in \text{Crit}(\mathfrak{F}_{in}) \setminus \text{Leaves}(\mathfrak{F}_{in})$. Moreover, we have that depth$_{\mathfrak{F}_{out}}(A) \geq \ell_{out}(S) - \ell_{out}(A) = \ell_{out}(S) - \ell_{out}(\Lambda_{out} \{S, S'\}) > \lambda$, where the second inequality follows from assertion (ii) of Lemma A.1. \hfill $\square$

Lemma A.2. Let $\mathfrak{F}_{in} = (T_{in}, \ell_{in})$ be an FCTS, let $\lambda > 0$, and let $\mathfrak{F}_{out} = (T_{out}, \ell_{out})$ be the FCTS that results from pruning $\mathfrak{F}_{in}$ by removing branches of length $\leq \lambda$ using
an \( \ell_{in} \)-increasing leaf enumeration \( \sigma = (\text{leaf}[1], \ldots, \text{leaf}[n]) \) of \( \text{Leaves}(T_{in}) \). Then:

(a) For all \( A \in \text{Nodes}(T_{in}) \setminus \text{Nodes}(T_{out}) \), \( A \uparrow T_{in} \cap \text{Nodes}(T_{out}) = \emptyset \).

(b) For all \( A \in \text{Nodes}(T_{in}) \), \( A \in \text{Nodes}(T_{out}) \) if, and only if, \( \text{lastLeaf}_\sigma(A, T_{in}) \in \text{Leaves}(T_{out}) \).

(c) For all \( A \in \text{Nodes}(T_{out}) \), \( \text{depth}_{T_{out}}(A) = \text{depth}_{T_{in}}(A) \).

Proof. For brevity, we write \( \text{lastLeaf}_\sigma(A) \) for \( \text{lastLeaf}_\sigma(A, T_{in}) \). Evidently, (a) follows from P4, and the “if” part of (b) follows from (a). To establish the “only if” part of (b), let \( A \in \text{Nodes}(T_{out}) \), and let \( \text{leaf}[i] = \text{lastLeaf}_\sigma(A) \). We need to show that \( \text{leaf}[i] \in \text{Nodes}(T_{out}) \). If \( i = n \) then this is true (by property P2), so let us assume \( i < n \). Let \( j \) be any element of the set \( \{i + 1, \ldots, n\} \) (that is, \( \text{leaf}[j] \notin \text{Leaves}(T_{in[A]}) \)).

Now we claim that \( \text{leaf}[j] \) must satisfy

\[
\ell_{in}(\text{leaf}[i]) - \ell_{in}(\bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[i]\}) > \lambda.
\]

To see this, let \( \text{leaf}[k] \) be any leaf of \( T_{out[A]} \); such a leaf must exist, by P4. As \( \text{leaf}[i] = \text{lastLeaf}_\sigma(A) \), we have that \( i \geq k \) and \( \ell_{in}(\text{leaf}[i]) \geq \ell_{in}(\text{leaf}[k]) \). As \( j \in \{i + 1, \ldots, n\} \), we have that \( j \in \{k + 1, \ldots, n\} \). Therefore, since \( \text{leaf}[k] \in \text{Leaves}(T_{out}) \), property P3 implies that:

\[
\ell_{in}(\text{leaf}[k]) - \ell_{in}(\bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[k]\}) > \lambda \quad (A.2)
\]

But, since \( \text{leaf}[i] \) and \( \text{leaf}[k] \) are leaves of \( T_{in[A]} \) but \( \text{leaf}[j] \) is not,

\[
\bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[i]\} = \bigwedge_{T_{in}} \{\text{leaf}[j], A\} = \bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[k]\}
\]

and (since \( \ell_{in}(\text{leaf}[i]) \geq \ell_{in}(\text{leaf}[k]) \)) this implies:

\[
\ell_{in}(\text{leaf}[i]) - \ell_{in}(\bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[i]\}) \geq \ell_{in}(\text{leaf}[k]) - \ell_{in}(\bigwedge_{T_{in}} \{\text{leaf}[j], \text{leaf}[k]\})
\]
This and Equation (A.2) imply that our claim is valid (for any \( j \in \{1, \ldots, n\} \)).

The “only if” part of (b) follows from this and property P3.

To prove (c), let \( A \in \text{Nodes}(T_{out}) \). Then \( \text{lastLeaf}_\sigma(A) \in \text{Leaves}(T_{out[A]}) \) (by (b)), and every \( W \in \text{Nodes}(T_{out[A]}) \subseteq \text{Nodes}(T_{in[A]}) \) satisfies \( \ell_{out}(W) = \ell_{in}(W) \leq \ell_{in}(\text{lastLeaf}_\sigma(A)) = \ell_{out}(\text{lastLeaf}_\sigma(A)) \).

It follows that \( \text{depth}_{\mathfrak{S}_{out}}(A) = \ell_{out}(\text{lastLeaf}_\sigma(A)) - \ell_{out}(A) = \ell_{in}(\text{lastLeaf}_\sigma(A)) - \ell_{in}(A) = \text{depth}_{\mathfrak{S}_{in}}(A) \). \( \Box \)

**Lemma A.3.** Let \( \mathfrak{S}_{in} = (T_{in}, \ell_{in}) \) be an FCTS, let \( \lambda > 0 \), and let \( \mathfrak{S}_{out} = (T_{out}, \ell_{out}) \) be the FCTS that results from pruning \( \mathfrak{S}_{in} \) by removing branches of length \( \leq \lambda \) using an \( \ell_{in}\)-increasing leaf enumeration \( \sigma = (\text{leaf}[1], \ldots, \text{leaf}[n]) \) of \( \text{Leaves}(T_{in}) \). Then:

(a) \( \text{Nodes}(T_{out}) \setminus \text{Leaves}(T_{out}) \supseteq U^\lambda \langle \mathfrak{S}_{in} \rangle \supseteq \text{Crit}(T_{out}) \setminus \text{Leaves}(T_{out}) \)

(b) For all \( A \in V^\lambda_1 \langle \mathfrak{S}_{in} \rangle \setminus V^\lambda_1 \langle \mathfrak{S}_{in} \rangle, A \upharpoonright T_{in} \cap \text{Nodes}(T_{out}) = \emptyset. \)

(c) For all \( A \in V^\lambda_1 \langle \mathfrak{S}_{in} \rangle, A \upharpoonright T_{in} \cap \text{Nodes}(T_{out}) = \text{Path}_\sigma(A, T_{in}). \)

**Proof.** For brevity, we shall write \( U^\lambda, V^\lambda_1, V^\lambda_1, \text{lastLeaf}_\sigma(A) \), and \( \text{Path}_\sigma(A, T_{in}) \) for \( U^\lambda \langle \mathfrak{S}_{in} \rangle, V^\lambda_1 \langle \mathfrak{S}_{in} \rangle, V^\lambda_1 \langle \mathfrak{S}_{in} \rangle, \text{lastLeaf}_\sigma(A, T_{in}), \) and \( \text{Path}_\sigma(A, T_{in}). \)

First, we prove (a). The inclusion \( U^\lambda \supseteq \text{Crit}(T_{out}) \setminus \text{Leaves}(T_{out}) \) follows from Corollary A.1 and Lemma A.2(c). Moreover, since P4 implies that \( \text{Leaves}(T_{out}) \subseteq \text{Leaves}(T_{in}) \), we have that \( U \notin \text{Leaves}(T_{out}) \) if \( U \in U^\lambda \). So the other inclusion of (a) will follow if we can show that \( U \in \text{Nodes}(T_{out}) \) whenever \( U \in U^\lambda \).

Let \( U \) be any element of \( U^\lambda \), and let \( \text{leaf}[i] = \text{lastLeaf}_\sigma(U) \). If \( i = n \), then \( \text{lastLeaf}_\sigma(U) \in \text{Nodes}(T_{out}) \) (by property P2) and so \( U \in \text{Nodes}(T_{out}) \) (because of P4), as required. Now suppose \( i < n \). Let \( j \) be any element of the set \( \{i+1, \ldots, n\} \) (so \( \text{leaf}[j] \notin \text{Leaves}(T_{in[U]}) \)). Since \( \text{leaf}[i] \) is a leaf of \( T_{in[U]} \) but \( \text{leaf}[j] \) is not, we have that \( \land_{T_{in}} \{\text{leaf}[j], \text{leaf}[i]\} \prec_{T_{in}} U. \) Hence:

\[
\ell_{in}(\text{leaf}[i]) - \ell_{in}(\land_{T_{in}} \{\text{leaf}[j], \text{leaf}[i]\}) > \ell_{in}(\text{leaf}[i]) - \ell_{in}(U) = \text{depth}_{\mathfrak{S}_{in}}(U) > \lambda
\]
We see from this and property P3 that \( \text{lastLeaf}_\sigma(U) = \text{leaf}[i] \in \text{Leaves}(T_{\text{out}}) \), and hence (in view of P4) that \( U \in \text{Nodes}(T_{\text{out}}) \), as required. This proves (a).

Next, we prove (b). Let \( A \) be any node in \( \mathbf{V}^\lambda \setminus \mathbf{V}_1^\lambda \). Then it follows from the definitions of \( \mathbf{V}^\lambda \) and \( \mathbf{V}_1^\lambda \) that \( A \neq \text{root}(T_{\text{in}}) \).

Let \( P = \text{parent}_{T_{\text{in}}}(A) \). Then \( P \in A \setminus T_{\text{in}} \subseteq \mathbf{U}^\lambda \), so we have that:

\[
\ell_{\text{in}}(\text{lastLeaf}_\sigma(P)) - \ell_{\text{in}}(P) = \text{depth}_{\sigma_{\text{in}}}(P) > \lambda
\]  
(A.3)

Now \( \ell_{\text{in}}(D) - \ell_{\text{in}}(A) \leq \text{depth}_{\sigma_{\text{in}}}(A) \) for all \( D \supseteq T_{\text{in}} A \). Therefore:

\[
\ell_{\text{in}}(D) - \ell_{\text{in}}(P) \leq \text{depth}_{\sigma_{\text{in}}}(A) + \ell_{\text{in}}(A) - \ell_{\text{in}}(P) \leq \lambda \quad \text{for all } D \supseteq T_{\text{in}} A \quad \text{(A.4)}
\]

Here the second inequality follows from the definition of \( \mathbf{V}_1^\lambda \) and the facts that \( P = \text{parent}_{T_{\text{in}}}(A) \) and \( A \in \mathbf{V}^\lambda \setminus \mathbf{V}_1^\lambda \). It follows from Equation (A.3) and Equation (A.4) that \( \text{lastLeaf}_\sigma(P) \) is not a descendant of \( A \) in \( T_{\text{in}} \), and so

\[
\wedge_{T_{\text{in}}} \{\text{lastLeaf}_\sigma(P), \text{lastLeaf}_\sigma(A)\} = P
\]  
(A.5)

Since \( \text{lastLeaf}_\sigma(A) \supseteq T_{\text{in}} A \), we deduce from Equation (A.4) and Equation (A.5) that

\[
\ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(\wedge_{T_{\text{in}}} \{\text{lastLeaf}_\sigma(P), \text{lastLeaf}_\sigma(A)\}) \leq \lambda
\]  
(A.6)

Since \( P = \text{parent}_{T_{\text{in}}}(A) \) and \( \text{lastLeaf}_\sigma(P) \neq \text{lastLeaf}_\sigma(A) \) (e.g., by Equation (A.5)), the leaf \( \text{lastLeaf}_\sigma(P) \) must occur later in the \( \ell_{\text{in}} \)-increasing enumeration \( \sigma \) than the leaf \( \text{lastLeaf}_\sigma(A) \). This, Equation (A.6), and P3 imply that \( \text{lastLeaf}_\sigma(A) \not\in \text{Leaves}(T_{\text{out}}) \). It now follows from assertion (b) of Lemma A.2 that \( A \not\in \text{Nodes}(T_{\text{out}}) \).

This and assertion (a) of Lemma A.2 imply \( A \upharpoonright T_{\text{in}} \cap \text{Nodes}(T_{\text{out}}) = \emptyset \), which proves
Finally, we prove (c). Let \( A \) be any node in \( V_1^\lambda \). We first make the claim that \( \text{lastLeaf}_\sigma(A) \) is a leaf of \( T_{\text{out}} \).

If \( A = \text{root}(T_{\text{in}}) \) then the claim is certainly true (by property P2), so let us assume \( A \neq \text{root}(T_{\text{in}}) \). Let \( P = \text{parent}_{T_{\text{in}}} (A) \), and let \( S \) be any leaf of \( T_{\text{in}} \) that occurs later in the \( \ell_{\text{in}} \)-increasing enumeration \( \sigma \) than \( \text{lastLeaf}_\sigma(A) \). Then \( S \notin \text{Leaves}(T_{\text{in}[A]}) \), and so \( \bigwedge_{T_{\text{in}}} \{ S, \text{lastLeaf}_\sigma(A) \} \preceq T_{\text{in}} P \), which implies that:

\[
\ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(\bigwedge_{T_{\text{in}}} \{ S, \text{lastLeaf}_\sigma(A) \}) \geq \ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(P)
\]

(A.7)

But, since depth\( \bar{\sigma}_{T_{\text{in}}}(A) = \ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(A) \), we also have that

\[
\ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(P) = \text{depth}_{\bar{\sigma}_{T_{\text{in}}}}(A) + \ell_{\text{in}}(A) - \ell_{\text{in}}(P) > \lambda
\]

(A.8)

where the inequality follows from the definition of \( V_1^\lambda \) and the facts that \( P = \text{parent}_{T_{\text{in}}} (A) \) and \( A \in V_1^\lambda \). Now it follows from Equation (A.7) and Equation (A.8) that:

\[
\ell_{\text{in}}(\text{lastLeaf}_\sigma(A)) - \ell_{\text{in}}(\bigwedge_{T_{\text{in}}} \{ S, \text{lastLeaf}_\sigma(A) \}) > \lambda
\]

Since this is true for every leaf \( S \) of \( T_{\text{in}} \) that occurs later in the \( \ell_{\text{in}} \)-increasing enumeration \( \sigma \) than \( \text{lastLeaf}_\sigma(A) \), our claim is justified (by property P3).

If \( W \) is any node in \( \text{Path}_\sigma(A) \), then \( W \in \text{lastLeaf}_\sigma(A) \downarrow T_{\text{in}} \) and so it follows from our claim (and P4) that \( W \in \text{Nodes}(T_{\text{out}}) \). Thus every node in \( \text{Path}_\sigma(A) \) lies in \( A \uparrow T_{\text{in}} \cap \text{Nodes}(T_{\text{out}}) \).

It remains only to prove that \( A \uparrow T_{\text{in}} \cap \text{Nodes}(T_{\text{out}}) \setminus \text{Path}_\sigma(A) = \emptyset \). To do this, we suppose there is a node \( X \in A \uparrow T_{\text{in}} \cap \text{Nodes}(T_{\text{out}}) \setminus \text{Path}_\sigma(A) \) and deduce a contradiction. As \( X \in A \uparrow T_{\text{in}} \setminus \text{Path}_\sigma(A) \), we have that \( X \notin \text{lastLeaf}_\sigma(A) \downarrow T_{\text{in}} \) and so \( \text{lastLeaf}_\sigma(A) \neq \text{lastLeaf}_\sigma(X) \). Moreover, each of the nodes \( \text{lastLeaf}_\sigma(X) \) and
lastLeaf_σ(A) is a leaf of T_{out} (by Lemma A.2(b) and our claim).

Let \( C = \bigwedge_{T_{out}} \{ \text{lastLeaf}_\sigma(X), \text{lastLeaf}_\sigma(A) \} \). Then we have that \( C \in \text{Crit}(T_{out}) \), \( c \notin \text{Leaves}(T_{out}) \), and \( C = \bigwedge_{T_{in}} \{ \text{lastLeaf}_\sigma(X), \text{lastLeaf}_\sigma(A) \} \) (by assertion (i) of Lemma A.1). The latter implies \( C \supseteq T_{in} A \) (as also \( \text{lastLeaf}_\sigma(X) \supseteq T_{in} X \supseteq T_{in} A \) and \( \text{lastLeaf}_\sigma(A) \supseteq T_{in} A \)); and \( C \supseteq T_{in} A \) implies \( \text{depth}_{\text{in}} C \leq \text{depth}_{\text{in}} A \leq \lambda \) (where the second inequality follows from the fact that \( A \in V_1^\lambda \subseteq V^\lambda \)). Hence \( C \not\subseteq U^\lambda \). But this contradicts assertion (a) (since \( C \in \text{Crit}(T_{out}) \setminus \text{Leaves}(T_{out}) \)). It follows that \( X \) cannot exist, and so our proof of (c) is complete. \( \square \)

We can now prove the main result of Sub-subsection 3.2.4.2:

**Proposition 5.** Let \( \mathcal{F}_{in} = (T_{in}, \ell_{in}) \) be an FCTS, let \( \lambda > 0 \), and let \( \mathcal{F}_{out} = (T_{out}, \ell_{out}) \) be the FCTS that results from pruning \( \mathcal{F}_{in} \) by removing branches of length \( \leq \lambda \) using an \( \ell_{in} \)-increasing enumeration \( \sigma \) of \( \text{Leaves}(T_{in}) \). Then the nodes of \( \mathcal{F}_{out} \) consist just of:

(i) The nodes of \( U^\lambda \langle \mathcal{F}_{in} \rangle \).

(ii) The nodes of \( \text{Path}_\sigma(A,T_{in}) \) for each node \( A \) in \( V_1^\lambda \langle \mathcal{F}_{in} \rangle \)

**Proof.** As \( U^\lambda \langle \mathcal{F}_{in} \rangle \subseteq \text{Nodes}(T_{out}) \) by Lemma A.3(a), on putting \( T = T_{in} \) and \( \mathcal{F} = \mathcal{F}_{in} \) in Equation (3.13) and taking the intersection of each side with \( \text{Nodes}(T_{out}) \) we see that:

\[
\text{Nodes}(T_{out}) = U^\lambda \langle \mathcal{F}_{in} \rangle \cup \bigcup_{A \in V_1^\lambda \langle \mathcal{F}_{in} \rangle} (A \uparrow_{T_{in}} \cap \text{Nodes}(T_{out}))
\]

The proposition follows from this and assertions (b) and (c) of Lemma A.3. \( \square \)
A.2 Properties of Simplification Step 3

Here we establish some properties of simplification step 3 that are used in our proof of the Theorem 4 and our justification of Algorithm 1.

For all \( j \in \{1, \ldots, |\mathcal{D}(\mathcal{F})|\} \), we see from E1–E5 that \( \text{Nodes}(\mathcal{F}^\text{crit}(\delta)) \subseteq \text{Nodes}(\mathcal{F}^\text{crit}(\delta')) \) whenever \( \delta \geq \delta' \). It follows that \( \mathcal{F}^\text{crit}(\cdot) \) has the following monotonicity property:

\[
\mathcal{F}^\text{crit}(\delta) \subseteq \mathcal{F}^\text{crit}(\delta') \text{ whenever } \delta \geq \delta'
\]  

(A.9)

In addition, \( \mathcal{F}^\text{crit}(\cdot) \) has the following four properties for every \( \lambda > 0 \) (as we will explain below):

E6: For every \( C \in \text{Nodes}(\mathcal{F}^\text{crit}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \) and every \( i \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\} \), \( C \in \text{Nodes}(\mathcal{F}^\text{crit}(d_{i+1})) \) if, and only if, for every \( j \in \{0, \ldots, i\} \), \( \ell(C) - \ell(\text{parent}_{\mathcal{F}^\text{crit}(d_{i+1})}(C)) > d_{i+1}^\mathcal{F} \).

E7: For every \( C \in \text{Nodes}(\mathcal{F}^\text{crit}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \),

\( C \in \text{Nodes}(\mathcal{F}^\text{crit}(\lambda)) \) if, and only if, there is no critical proper ancestor \( C' \) of \( C \) in \( \mathcal{F} \) such that \( \ell(C) - \ell(C') \leq \lambda \) and \( C' \in \text{Nodes}(\mathcal{F}^\text{crit}(\ell(\text{LCN}(\mathcal{F}))) \cup \{\text{root}(\mathcal{F})\}) \).

E8: For every \( C \in \text{Nodes}(\mathcal{F}^\text{crit}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \),

\( C \in \text{Nodes}(\mathcal{F}^\text{crit}(\lambda)) \) if \( \ell(C) - \ell(\text{parent}_{\mathcal{F}^\text{crit}(\lambda)}(C)) > \lambda \).

E9: For every \( C \in \text{Nodes}(\mathcal{F}^\text{crit}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \), if \( C \in \text{Nodes}(\mathcal{F}^\text{crit}(\lambda)) \) then \( \ell(C) - \ell(\text{parent}_{\mathcal{F}^\text{crit}(\lambda)}(C)) > \lambda \).

Our proof of the correctness of Algorithm 1 will be based on property E7. However, E1–E3, E8, and E9 are the only properties of simplification step 3 that will be used in our proof of the Theorem 4.
E6 is easily deduced from E5 by induction on $i$. Now we establish E7 – E9. Let $C \in \text{Nodes}(\mathcal{F}^{\text{crit}}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\})$, and let $\lambda$ be any positive value. We first claim that, for any critical proper ancestor $C'$ of $C$ in $\mathcal{F}$, the following four conditions are equivalent:

(a) There is some $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$ such that $\ell(C) - \ell(C') \leq d^\mathcal{F}_{j+1} \leq \lambda$ and $C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(d^\mathcal{F}_j))$.

(b) There is some $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$ such that $\ell(C) - \ell(C') \leq d^\mathcal{F}_{j+1} \leq \lambda$ and $C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\text{pred}_\mathcal{F}(\ell(C) - \ell(C'))))$.

(c) $\ell(C) - \ell(C') \leq \lambda$ and $C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\text{pred}_\mathcal{F}(\ell(C) - \ell(C'))))$.

(d) There is some $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$ such that $\ell(C) - \ell(C') = d^\mathcal{F}_{j+1} \leq \lambda$ and $C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(d^\mathcal{F}_j))$.

Here (a) implies (b) because of the monotonicity property Equation (A.9) and the fact that if $\ell(C) - \ell(C') \leq d^\mathcal{F}_{j+1}$ then $\text{pred}_\mathcal{F}(\ell(C) - \ell(C')) \leq d^\mathcal{F}_j$. Evidently, (b) implies (c), and (d) implies (a). For any critical proper ancestor $C'$ of $C$ in $\mathcal{F}$, $\ell(C) - \ell(C') = d^\mathcal{F}_{j+1}$ and $\text{pred}_\mathcal{F}(\ell(C) - \ell(C')) = d^\mathcal{F}_j$ for some $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$, and so (c) implies (d). This justifies our claim that (a)–(d) are equivalent.

Next, we observe that $C \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda))$ holds if, and only if, $C$ satisfies $\ell(C) - \ell(\text{parent}_{\mathcal{F}^{\text{crit}}}(d^\mathcal{F}_j)) \geq d^\mathcal{F}_{j+1}$ for all $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$ such that $d^\mathcal{F}_{j+1} \leq \lambda$. (This follows from E6 when $\lambda \in \mathcal{D}(\mathcal{F})$. It remains true if $\lambda \notin \mathcal{D}(\mathcal{F})$, because of E4.) So $C \notin \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda))$ just if there is some $j \in \{0, \ldots, |\mathcal{D}(\mathcal{F})| - 1\}$ such that $\ell(C) - \ell(\text{parent}_{\mathcal{F}^{\text{crit}}}(d^\mathcal{F}_j)) \leq d^\mathcal{F}_{j+1} \leq \lambda$. Thus $C \notin \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda))$ just if (a) holds for some critical proper ancestor $C'$ of $C$ in $\mathcal{F}$. Equivalently, $C \notin \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda))$ just if (c) holds for some critical proper ancestor $C'$ of $C$ in $\mathcal{F}$. This proves E7. E8 follows from the “if” part of E7.

Suppose the node $C$ violated E9. Then $C \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda))$. Moreover, when $C' = \text{parent}_{\mathcal{F}^{\text{crit}}}(\lambda)(C)$ we would have that $\ell(C) - \ell(C') \leq \lambda$ and also that
\( C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\text{pred}_\mathcal{F}(\ell(C) - \ell(C')))) \), where the latter follows from the former, the fact that \( C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\lambda)) \), and the monotonicity property (A.9). But this would contradict the “only if” part of E7. So E9 holds.

### A.3 Justification of Algorithm 1

The correctness of Algorithm 1 will be deduced from Lemma A.4 and Corollary A.2 below.

Let \( \mathcal{F} = (T, \ell) \) be any FCTS, and let \( C \) be any node of \( \mathcal{F}^{\text{crit}} \). Then we define

\[
\delta_\lambda(C, \mathcal{F}) = \begin{cases} 
\infty & \text{if } C \in \text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}, \\
\ell(C) - \ell(Q_\lambda(C, \mathcal{F})) & \text{otherwise},
\end{cases}
\]

where \( Q_\lambda(C, \mathcal{F}) \) is the closest critical proper ancestor \( C_0 \) of \( C \) in \( \mathcal{F} \) such that

- \( \ell(C) - \ell(C_0) > \lambda \)
- \( \ell(C) - \ell(C_0) \leq \lambda \) and \( C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\text{pred}_\mathcal{F}(\ell(C) - \ell(C')))) \)

\( Q_\lambda(C, \mathcal{F}) \) exists for all \( C \in \text{Nodes}(\mathcal{F}^{\text{crit}}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \), because when \( C' = \text{LCN}(\mathcal{F}) \) we see from E2 that \( C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\mu)) \) for every \( \mu \geq 0 \) and so \( C' \) must satisfy the “either” or the “or” condition. Now \( \delta_\lambda(\cdot, \mathcal{F}) \) satisfies the following condition:

**Lemma A.4.** Let \( 0 \leq \mu \leq \lambda \) and let \( \mathcal{F} = (T, \ell) \) be any FCTS. Then for all \( C \in \text{Nodes}(\mathcal{F}^{\text{crit}}) \) we have that \( \delta_\lambda(C, \mathcal{F}) > \mu \) if, and only if, \( C \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\mu)) \).

**Proof.** Suppose \( C \in \text{Nodes}(\mathcal{F}^{\text{crit}}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}) \). Then \( \delta_\lambda(C, \mathcal{F}) > \mu \) holds just if \( \ell(C) - \ell(Q_\lambda(C, \mathcal{F})) > \mu \), and since \( \mu \leq \lambda \) we see from the definition of \( Q_\lambda(C, \mathcal{F}) \) that this holds just if no critical proper ancestor \( C' \) of \( C \) in \( \mathcal{F} \) satisfies \( \ell(C) - \ell(C') \leq \mu \) and \( C' \in \text{Nodes}(\mathcal{F}^{\text{crit}}(\text{pred}_\mathcal{F}(\ell(C) - \ell(C')))) \). So, in this case, the lemma follows from E7.
The lemma also holds if $C \in \text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\}$, because in that case $\delta(C, \mathcal{F}) = \infty > \mu$ and E1 – E3 imply $C \in \text{Nodes}(\mathcal{F}_{\text{crit}}(\mu))$. \hfill \Box

**Corollary A.2.** Let $\lambda$ be any positive value, let $\mathcal{F} = (T, \ell)$ be any FCTS, and let $C \in \text{Nodes}(\mathcal{F}_{\text{crit}}) \setminus (\text{Leaves}(\mathcal{F}) \cup \{\text{LCN}(\mathcal{F})\} \cup \{\text{root}(\mathcal{F})\})$. Then $\delta(C, \mathcal{F}) = \ell(C) - \ell(Q)$, where $Q$ is the closest critical proper ancestor $C'$ of $C$ in $\mathcal{F}$ such that

- either $\ell(C) - \ell(C') > \lambda$
- or $\ell(C) - \ell(C') \leq \lambda$ and $\ell(C) - \ell(C') \leq \delta(C', \mathcal{F})$

**Proof.** We just have to show that $Q = Q_{\lambda}(C, \mathcal{F})$. The definition of $Q_{\lambda}(C, \mathcal{F})$ differs from the definition of $Q$ only in the or condition “$\ell(C) - \ell(C') \leq \lambda$ and $C' \in \text{Nodes}(\mathcal{F}_{\text{crit}}(\text{pred}_{\mathcal{F}}(\ell(C) - \ell(C'))))$”.

On putting $\mu = \text{pred}_{\mathcal{F}}(\ell(C) - \ell(C'))$ in Lemma A.4, we see that this condition holds if, and only if, $\ell(C) - \ell(C') \leq \lambda$ and $\text{pred}_{\mathcal{F}}(\ell(C) - \ell(C')) < \delta(C', \mathcal{F})$, which is equivalent to the or condition in the definition of $Q$ (because either $\delta(C', \mathcal{F}) = \ell(C) - \ell(Q_{\lambda}(C', \mathcal{F})) \in D(\mathcal{F})$ or $\delta(C', \mathcal{F}) = \infty$). So $Q = Q_{\lambda}(C, \mathcal{F})$, as required. \hfill \Box

We can now explain why Algorithm 1 is correct. The algorithm sets $(T, \ell)$ to a clone of $\mathcal{F}_{\text{crit}} = (T_{\text{crit}}, \ell_{\text{crit}})$. Writing $\mathcal{F}$ for $(T, \ell)$, we claim that the label $\text{label}$ given by the algorithm to each node $C$ of $\mathcal{F} = \mathcal{F}_{\text{crit}}$ is just the value $\delta(C, \mathcal{F})$. Assuming this claim is valid, the correctness of the algorithm follows from Lemma A.4. So it remains only to verify the claim.

The claim is certainly valid if $C$ is $\text{root}(\mathcal{F})$ or $\text{LCN}(\mathcal{F})$, because those nodes are given the label $\infty$.

We see that the algorithm does a top-down traversal of $T[\text{LCN}(\mathcal{F})]$, during which the procedure $\text{labelDescendants}$ is executed once for each proper descendant $C$ of $\text{LCN}(\mathcal{F})$ in $\mathcal{F}$. When $\text{labelDescendants}$ is executed for such a node $C$ that is a leaf, it gives $C$ the label $\infty$. So the claim is valid for each proper descendant $C$ of $\text{LCN}(\mathcal{F})$ that is a leaf.
When \texttt{labelDescendants} is executed for a proper descendant $C$ of $\textbf{LCN}(\mathcal{F})$ that is not a leaf, the \texttt{repeat} loop in the procedure is executed. It follows from Corollary A.2 that this loop labels $C$ with the value $\delta_{\lambda}(C, \mathcal{F})$. (Note that, when the loop is executed, $C'.\text{label} = \delta_{\lambda}(C', \mathcal{F})$ for each proper ancestor $C'$ of $C$ in $\mathcal{F}$.) Therefore the claim is also valid for each proper descendant $C$ of $\textbf{LCN}(\mathcal{F})$ that is not a leaf.

Thus the claim is valid for all nodes $C$ of $\mathcal{F} = \mathcal{F}^{\text{crit}}$, and Algorithm 1 is correct.
Appendix B

A Constructive Proof of Theorem 4

For any adjacency relation $\pi$, any digital picture $(V, \pi, f)$, any $\lambda > 0$, and any integer $k \geq 0$, let us say that the digital picture $(V, \pi, f)$ is $(\lambda, k)$-good with respect to $\pi$ if $\Lambda_{(V, \pi, f)} > \lambda$ and $K_{(V, \pi, f)} > k$. Also, let us say that a digital picture $(V, \pi, f')$ is an $\varepsilon$-perturbation of a digital picture $(V, \pi, f)$ if $(V, \pi, f')$ has $\|f' - f\|_{\infty} \leq \varepsilon$. Then Theorem 4 can be deduced from the following lemma:

**Lemma B.1.** Let $\pi$ be any adjacency relation and $(V, \pi, f_{\text{good}})$ a digital picture. Let $\varepsilon$ be a positive value, let $k$ be a nonnegative integer for which $(V, \pi, f_{\text{good}})$ is $(4\varepsilon, k)$-good with respect to $\pi$, and let $(V, \pi, f')$ be an $\varepsilon$-perturbation of $(V, \pi, f)_{\text{good}}$. Then there is an essential isomorphism of $\text{FCTS}_{(V, \pi, f_{\text{good}})}$ to the $(2\varepsilon, k)$-simplification of $\text{FCTS}_{(V, \pi, f')}$. That is level-preserving to within $\varepsilon$.

**Proof.** of the Theorem 4 assuming Lemma B.1 is valid: Suppose $(V, \pi, f), \lambda$, and $k$ satisfy the hypotheses of Theorem 4, so that $0 < \lambda < \Lambda_{(V, \pi, f)}/2$ and $0 \leq k < K_{(V, \pi, f)}$. Let $(V, \pi, f')$ be any digital picture that satisfies the conditions stated in the theorem (i.e., let $(V, \pi, f')$ be any digital picture whose domain is the same as that of $(V, \pi, f)$ and which satisfies the condition $\|f' - f\|_{\infty} \leq \lambda/2$). Then we need to show that the conclusion of Theorem 4 holds—i.e., that there is an essential isomorphism of the $(\lambda, k)$-simplification of $\text{FCTS}_{(V, \pi, f')}$ to $\text{FCTS}_{(V, \pi, f)}$ that is level-preserving.
to within $\lambda/2$. We now deduce this from the Lemma B.1.

Let $(V, \pi, f_{good}) = (V, \pi, f)$, and let $\varepsilon = \lambda/2$. Then $4\varepsilon = 2\lambda < \Lambda_{(V,\pi,f_{good})} = \Lambda_{(V,\pi,f)}$ and $k < K_{(V,\pi,f_{good})} = K_{(V,\pi,f)}$, so that $(V, \pi, f_{good})$ is $(4\varepsilon,k)$-good with respect to $\pi$. We also have that $\|f' - f_{good}\|_\infty = \|f' - f\|_\infty \leq \lambda/2 = \varepsilon$, so that $(V, \pi, f')$ is a $\varepsilon$-perturbation of $(V, \pi, f_{good})$. Thus $(V, \pi, f_{good}) = (V, \pi, f)$ and $(V, \pi, f')$ satisfy the hypotheses of the Lemma B.1, and must therefore satisfy the conclusion of the lemma, which implies the conclusion of Theorem 4 since $2\varepsilon = \lambda$.

We now prove the Lemma B.1 by constructing an explicit essential isomorphism of $\text{FCTS}_{(V,\pi,f_{good})}$ to the $(2\varepsilon,k)$-simplification of $\text{FCTS}_{(V,\pi,f')}$ that is level-preserving to within $\varepsilon$.

Let $\mathfrak{F}_{good} = (T_{good}, \ell_{good}) = \text{FCTS}_{(V,\pi,f_{good})}$, and let $\mathfrak{F}' = (T', \ell') = \text{FCTS}_{(V,\pi,f')}$. Let $\mathfrak{F}_1 = (T_1, \ell_1)$ be the FCTS that results from pruning $\mathfrak{F}'$ by removing nodes of size $\leq k$, and let $(V, \pi, f_1)$ be the digital picture $(V, \pi, f)\mathfrak{F}_1$, so that $\mathfrak{F}_1 = \text{FCTS}_{(V,\pi,f_1)}$. Let $\mathfrak{F}_2 = (T_2, \ell_2)$ be the FCTS that results from pruning $\mathfrak{F}_1$ by removing branches of length $\leq 2\varepsilon$, and let $\mathfrak{F}_3 = (T_3, \ell_3)$ be the FCTS that results from eliminating internal edges of length $\leq 2\varepsilon$ from $\mathfrak{F}_2^{\text{crit}}$. Then $\mathfrak{F}_3 = (T_3, \ell_3)$ is the $(2\varepsilon,k)$-simplification of $\text{FCTS}_{(V,\pi,f')}$, so what we want to do is to construct an essential isomorphism of $\mathfrak{F}_{good}$ to $\mathfrak{F}_3$ that is level-preserving to within $\varepsilon$. We will do this in three steps:

**Step 1:** We define a suitable mapping $\phi : \text{Leaves}(T_{good}) \rightarrow \text{Leaves}(T_1)$.

**Step 2:** We show that $\phi$ is 1-to-1, and that the range of the mapping $\phi$ is exactly the set of all the leaves of the subtree $T_2$ of $T_1$. Thereafter, we regard $\phi$ as a bijection $\phi : \text{Leaves}(T_{good}) \rightarrow \text{Leaves}(T_2)$.

**Step 3:** We extend $\phi$ to a mapping $\varphi : \text{Crit}(T_{good}) \rightarrow \text{Crit}(T_2)$ by defining $\varphi(U) = \bigwedge_{T_2} \phi [\text{Leaves}(T_{good}[U])]$. We then establish that, for all $U, U' \in \text{Crit}(T_{good})$,
\( \varphi(U) \preceq_{T_2} \varphi(U') \) if, and only if, \( U \preceq_{\text{good}} U' \), so that \( \varphi \) is 1-to-1 and order-preserving. We also show that the range of \( \varphi \) is the subset \( \text{Crit}(T_3) \) of \( \text{Crit}(T_2) \), and that \( |\ell_3(\varphi(U)) - \ell_{\text{good}}(U)| \leq \varepsilon \) for every \( U \in \text{Crit}(T_{\text{good}}) \). Hence we can regard \( \varphi \) as a mapping \( \varphi : \text{Crit}(T_{\text{good}}) \to \text{Crit}(T_3) \) and, when so regarded, \( \varphi \) is an essential isomorphism of \( \bar{\mathcal{T}}_{\text{good}} \) to \( \bar{\mathcal{T}}_3 \) that is level-preserving to within \( \varepsilon \).

Note that the extension of \( \phi \) to \( \varphi \) in step 3 is very natural because, if \( T \) is any rooted tree and \( U \in \text{Crit}(T) \), then \( U = \bigwedge_T \text{Leaves}(T_{[U]}) \). (In fact \( U \in \text{Crit}(T) \) if, and only if, \( U \in \text{Nodes}(T) \) and \( U = \bigwedge_T \text{Leaves}(T_{[U]}) \).)

**B.1 Step 1 of the Proof of the Lemma B.1**

We begin by defining a class of symmetric and transitive relations (on spels) that will be used in our definition of the mapping \( \phi \).

Let \((V, \pi, f)\) be a digital picture, let \( \tau \in \mathbb{R} \), and let \( s \in V \). Then \( C_{(V, \pi, f)}(s, \tau) \) will denote the set of all \( s' \in V \) for which there exists a \( \pi \)-path \( s_0, \ldots, s_l \) such that \( s_0 = s, s_l = s' \), and \( f(s_i) \geq \tau \) for \( 0 \leq i \leq l \). Note that \( C_{(V, \pi, f)}(s, \tau) = \emptyset \) if \( \tau > f(s) \), and \( s \in C_{(V, \pi, f)}(s, \tau) \) if \( \tau \leq f(s) \).

If \((V, \pi, f)\) is a digital picture and \( \tau \in \mathbb{R} \), then we write \( s \equiv_{(V, \pi, f)} \tau \Rightarrow t \) to mean that \( s, t \in V \) and \( t \in C_{(V, \pi, f)}(s, \tau) \). It is readily confirmed that \( \equiv_{(V, \pi, f)} \Rightarrow \) is a symmetric and transitive relation (which depends on \( \pi \)), and that \( s \equiv_{(V, \pi, f)} \Rightarrow s \) if, and only if, \( f(s) \geq \tau \). Moreover, if \( s \equiv_{(V, \pi, f)} \tau_1 \Rightarrow t \) and \( t \equiv_{(V, \pi, f)} \tau_2 \Rightarrow u \) then \( s \equiv_{(V, \pi, f)} \min(\tau_1, \tau_2) \Rightarrow u \).

Now let \( C_{(V, \pi, f_{\text{good}})}(v) \) be any leaf of \( T_{\text{good}} \), and let \( z \) be any spel such that

\[
z \in \arg\min_{u \equiv_{(V, \pi, f_{\text{good}})} \leq f_{\text{good}}(v) - 2\varepsilon \Rightarrow v} f_1(u)
\]  

(B.1)
It follows from Equation (B.1) that:

\[ C(V, \pi, f_1)(z) \supseteq \{ u \mid u \in (V, \pi, f)_{\text{good}} \geq (V, \pi, f)_{\text{good}}(v) - 2\epsilon \Rightarrow v \} = C(V, \pi, f)_{\text{good}}(v, f)_{\text{good}}(v) - 2\epsilon \]

(B.2)

Next, we define:

\[ M(C(V, \pi, f)_{\text{good}}(v)) = \text{Leaves}(T_1 | C(V, \pi, f)(z)) \]

(B.3)

The set \( M(C(V, \pi, f)_{\text{good}}(v)) \) is well defined by Equation (B.3) for the following reasons. First, if \( v' \) is any spel such that \( C(V, \pi, f)_{\text{good}}(v') = C(V, \pi, f)_{\text{good}}(v) \) (so that \( f(v') = f(v) \)) then the condition obtained from Equation (B.1) when we replace \( v \) with \( v' \) is equivalent to Equation (B.1). Second, if \( z' \) is any spel that belongs to the set in Equation (B.1), then \( C(V, \pi, f_1)(z') = C(V, \pi, f_1)(z) \) (since \( f_1(z') = f_1(z) \)), and Equation (B.2) implies \( z' \in C(V, \pi, f_1)(z) \).

We can now define the mapping \( \phi : \text{Leaves}(T) \rightarrow \text{Leaves}(T_1) \) by defining \( \phi(C(V, \pi, f)_{\text{good}}(v)) \) to be the element of \( M(C(V, \pi, f)_{\text{good}}(v)) \) that occurs later in the \( \ell_1 \)-increasing leaf enumeration that is used in pruning \( (T_1, \ell_1) \) (to produce \( (T_2, \ell_2) \)) than all other elements of \( M(C(V, \pi, f)_{\text{good}}(v)) \). Note that if \( M(C(V, \pi, f)_{\text{good}}(v)) \) has just one element, then \( \phi(C(V, \pi, f)_{\text{good}}(v)) \) is that element.

This completes step 1 of the proof of the Lemma B.1.

### B.2 Some Useful Observations

Steps 2 and 3 of the proof of Lemma B.1 will be based on the following observations:

A. If \( (T, \ell) = \text{FCTS}_{(V, \pi, f)} \), where \( (V, \pi, f) \) is an arbitrary digital picture and \( \emptyset \neq S \subseteq \text{Nodes}(T) \), then \( \ell(\bigwedge_T S) \) is the greatest real value \( \tau \) such that \( s \in (V, \pi, f) \geq \tau \Rightarrow t \) for all spels \( s, t \in \bigcup S \).
B. Whenever $\emptyset \neq \mathbf{L} \subseteq \mathbf{L}' \subseteq \text{Leaves}(T_{\text{good}})$ and $\wedge_{T_{\text{good}}} \mathbf{L}' \neq \wedge_{T_{\text{good}}} \mathbf{L}$, we have that
\[
\ell_{\text{good}}(\wedge_{T_{\text{good}}} \mathbf{L}') < \ell_{\text{good}}(\wedge_{T_{\text{good}}} \mathbf{L}) - 4\varepsilon.
\]

C. If $v \in A \in \text{Leaves}(T_{\text{good}})$, $U \in \text{Nodes}(T_{\text{good}})$, and $A \nsubseteq_{\text{good}} U$, then we have that
\[
\ell_{\text{good}}(\wedge_{T_{\text{good}}} \{U, A\}) < \ell_{\text{good}}(A) - 4\varepsilon = f_{\text{good}}(v) - 4\varepsilon.
\]

D. If $C_{(V, \pi, f_{\text{good}})}(v) \in \text{Leaves}(T_{\text{good}})$ and $u \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 4\varepsilon \Rightarrow v$, then we have that $u \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(u) \Rightarrow v$ or, equivalently, $C_{(V, \pi, f_{\text{good}})}(u) \supseteq C_{(V, \pi, f_{\text{good}})}(v)$.

E. If $C_{(V, \pi, f_1)}(x) \in \text{Leaves}(T_1)$, then $C_{(V, \pi, f_1)}(x) \in \text{Leaves}(T_2)$ if, and only if, there is no node $C_{(V, \pi, f_1)}(y) \in \text{Leaves}(T_1)$ that satisfies both of the following conditions:

(i) $x \in (V, \pi, f_1) \geq f_1(x) - 2\varepsilon \Rightarrow y$

(ii) The leaf $C_{(V, \pi, f_1)}(y)$ occurs later in the $\ell_1$-increasing leaf enumeration that is used in pruning $(T_1, \ell_1)$ to produce $(T_2, \ell_2)$ than the leaf $C_{(V, \pi, f_1)}(x)$.

Here $A$ is a consequence of the definitions of $\text{FCTS}_{(V, \pi, f)}$ and $\wedge_T S$. (The special case of $A$ in which $S \subseteq \text{Leaves}(T)$ is of particular interest; note that in this case $s \in \bigcup S$ if, and only if, $C_{(V, \pi, f)}(s) \in S$.) $B$ is a consequence of the fact that $\Lambda_{(V, \pi, f_{\text{good}})} > 4\varepsilon$, $C$ can be deduced from $B$ by putting $\mathbf{L} = \{A\}$ and $\mathbf{L}' = \{A\} \cup \text{Leaves}(T_{\text{good}}[U])$, and $D$ can be deduced from $A$ and $C$.

Assertion $E$ is a consequence of $A$ and the fact that $(T_2, \ell_2)$ is the result of pruning $(T_1, \ell_1)$ by removing branches of length $\leq 2\varepsilon$. In view of assertion (ii) of Lemma A.1, we also have the following related fact:

$E'$. $\ell_1(\wedge_{T_1} \{Z, Z'\}) < \min(\ell_1(Z), \ell_1(Z')) - 2\varepsilon$ whenever $Z$ and $Z'$ are distinct leaves of $T_2$.

We could of course replace $\ell_1$ with $\ell_2$ in $E'$. Moreover, in view of assertion (i) of Lemma A.1, we could also replace $\wedge_{T_1}$ with $\wedge_{T_2}$. 
Now let \( x \) be any spel in \( V \). As \( \overline{\text{S}}_1 \) is the result of pruning \( \text{FCTS}_{(V, \pi, f')} = (T', \ell') \) by removing nodes of size \( \leq k \), and \( (V, \pi, f_1) = (V, \pi, f)_{\overline{\text{S}}_1} \), we see from the definition of \( (V, \pi, f)_{\overline{\text{S}}_1} \) that \( f_1(x) = \max \{ \ell'(U) \mid U \in \text{Nodes}(T'), |U| \geq k + 1, \ x \in U \} \). This is equivalent to

\[
 f_1(x) = \max \{ f'(y) \mid y \in V, x \in C_{(V, \pi, f')}(y), \text{ and } |C_{(V, \pi, f')}(y)| \geq k + 1 \} \tag{B.4}
\]

since the nodes \( U \in \text{Nodes}(T') \) for which \( x \in U \) are just the sets \( C_{(V, \pi, f')}(y) \) for which \( x \in C_{(V, \pi, f')}(y) \). Now we claim that:

\[
 f_1(x) = \max \{ \tau \mid |C_{(V, \pi, f')}(x, \tau)| \geq k + 1 \} \tag{B.5}
\]

To see this, we first observe that if \( y \) satisfies \( x \in C_{(V, \pi, f')}(y) \) then \( y \) also satisfies \( C_{(V, \pi, f')}(y) = C_{(V, \pi, f')}(x, f'(y)) \). It follows from this observation that each element of the set \( \{ f'(y) \mid y \in V, x \in C_{(V, \pi, f')}(y), \text{ and } |C_{(V, \pi, f')}(y)| \geq k + 1 \} \) in Equation (B.4) belongs to the set \( \{ f'(y) \mid y \in V \text{ and } |C_{(V, \pi, f')}(x, f'(y))| \geq k + 1 \} \) and therefore belongs to the set \( \{ \tau \mid |C_{(V, \pi, f')}(x, \tau)| \geq k + 1 \} \) in our claim Equation (B.5). So the right side of Equation (B.5) is no less than the right side of Equation (B.4); it remains to show that it is no greater.

For every \( \tau \leq f'(x) \), let \( y(\tau, x) \) be any spel in \( \arg\min_{s \in C_{(V, \pi, f')}(x, \tau)} f'(s) \), so that \( f'(y(\tau, x)) \geq \tau \), and it is easy to see that

\[
 C_{(V, \pi, f')}(y(\tau, x)) = C_{(V, \pi, f')}(x, \tau) \tag{B.6}
\]

since \( (V, \pi, f') \geq f'(y(\tau, x)) \) at every spel in \( C_{(V, \pi, f')}(x, \tau) \). Now if \( \tau_0 \) is any element of the set \( \{ \tau \mid |C_{(V, \pi, f')}(x, \tau)| \geq k + 1 \} \), then we have that \( f'(y(\tau_0, x)) \geq \tau_0 \) and we see from Equation (B.6) that \( |C_{(V, \pi, f')}(y(\tau_0, x))| \geq k + 1 \) and \( x \in C_{(V, \pi, f')}(y(\tau_0, x)) \), so that \( f(y(\tau_0, x)) \) is an element of \( \{ f'(y) \mid y \in V, x \in C_{(V, \pi, f')}(y), \text{ and } |C_{(V, \pi, f')}(y)| \geq k + 1 \} \).
The main goals of this step are to show that the mapping $B.3$ Step 2 of the Proof of the Lemma B.1
e deduced from (i) by putting an
On the other hand, whenever $K$ putting (B.4) and Equation (B.5) are equal, and so our claim Equation (B.5) follows from Equation (B.4).

Next, we establish the following properties of $(V, \pi, f_1)$:

F. $(V, \pi, f_1)$ is an $\varepsilon$-perturbation of $(V, \pi, f_{good})$, and if $((V, \pi, f_1), (V, \pi, f_b)) = ((V, \pi, f_1), (V, \pi, f_{good}))$ or $((V, \pi, f_{good}), (V, \pi, f_1))$ then for any $\tau, \delta \in \mathbb{R}$ and any spels $s, t, u \in V$ we have that:

(i) If $s \ll (V, \pi, f_a) \geq \tau \Rightarrow t$ then $s \ll (V, \pi, f_b) \geq \tau - \varepsilon \Rightarrow t$.

(ii) If $s \ll (V, \pi, f_a) \geq f_a(u) - \delta \Rightarrow t$ then $s \ll (V, \pi, f_b) \geq f_b(u) - \delta - 2\varepsilon \Rightarrow t$.

To see that $(V, \pi, f_1)$ has these properties, let $x$ be any spel in $V$ and note that $C_{(V, \pi, f_{good})}(x, \tau) \subseteq C_{(V, \pi, f_{good})}(x, \tau - \varepsilon)$ for every $\tau \in \mathbb{R}$ since $\|f' - f_{good}\|_\infty \leq \varepsilon$. On putting $\tau = f_{good}(x)$, we deduce that $C_{f_p}(x, f_{good}(x) - \varepsilon) \supseteq C_{(V, \pi, f_{good})}(x, f_{good}(x)) = C_{(V, \pi, f_{good})}(x)$, whence $|C_{(V, \pi, f_{good})}(x, f_{good}(x) - \varepsilon)| \geq |C_{(V, \pi, f_{good})}(x)| \geq k + 1$ (as $K_{(V, \pi, f_{good})} > k$). It follows from this and (B.5) that $(V, \pi, f_1)(x) \geq f_{good}(x) - \varepsilon$.

On the other hand, whenever $\tau > f_{good}(x) + \varepsilon$ we have that $f'(x) < \tau$ (as $\|f' - f_{good}\|_\infty \leq \varepsilon$), which implies that $|C_{(V, \pi, f')} (x, \tau)| = 0$ and hence (by (B.5)) that $f_1(x) < \tau$. From this it follows that $f_1(x) \leq f_{good}(x) + \varepsilon$. This shows that $f_1$ is an $\varepsilon$-perturbation of $f_{good}$, as $F$ asserts. Now (i) follows immediately, and (ii) can be deduced from (i) by putting $\tau = f_a(u) - \delta$, since the fact that $(V, \pi, f_a)$ is an $\varepsilon$-perturbation of $(V, \pi, f_b)$ implies that $f_a(u) - \delta \geq f_b(u) - \delta - \varepsilon$ for every $u \in V$.

**B.3 Step 2 of the Proof of the Lemma B.1**

The main goals of this step are to show that the mapping $\phi$ defined in step 1 of the proof is 1-to-1 and that the range of $\phi$ is exactly the subset Leaves($T_2$) of
\textbf{Leaves}(T_1). This will allow us to regard \( \phi \) as a bijection \( \phi : \text{Leaves}(T_{\text{good}}) \to \text{Leaves}(T_2) \).

We first state and prove the following easy lemma:

\textbf{Lemma B.2.} Let \( C(V, \pi, f_{\text{good}})(v) \) be any leaf of \( T_{\text{good}} \). Let \( x \) be any spel in \( V \) that satisfies \( x \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 2\varepsilon \equiv v \), and let \( S \) be any leaf of \( T_1 \) such that \( S \geq T_1 C(V, \pi, f_1)(x) \). Then \( S \in \mathcal{M}(C(V, \pi, f_{\text{good}})(v)) \).

\textbf{Proof.} Let \( z \) be a spel that satisfies Equation (B.1) with respect to \( v \). Then Equation (B.2) implies that \( x \in C(V, \pi, f_1)(z) \) and hence that \( C(V, \pi, f_1)(x) \geq T_1 C(V, \pi, f_1)(z) \). This and Equation (B.3) imply \( S \in \mathcal{M}(C(V, \pi, f_{\text{good}})(v)) \). \( \square \)

Next, we establish the following properties of \( \mathcal{M} \) and the mapping \( \phi \):

G. The following are true for any leaf \( C(V, \pi, f_{\text{good}})(v) \) of \( T_{\text{good}} \):

(a) If \( C(V, \pi, f_1)(y) \in \mathcal{M}(C(V, \pi, f_{\text{good}})(v)) \), then:

(i) \( y \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 4\varepsilon \equiv v \)

(ii) \( y \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(y) \equiv v \)

(iii) \( y \in (V, \pi, f_1) \geq f_1(y) - 2\varepsilon \equiv v \)

(b) If \( C(V, \pi, f_1)(y) = \phi(C(V, \pi, f_{\text{good}})(v)) \), then:

(i) \( f_{\text{good}}(v) + \varepsilon \geq f_1(y) \geq f_1(v) \geq f_{\text{good}}(v) - \varepsilon \)

(ii) \( y \in f_{\text{good}} \geq f_{\text{good}}(v) - 2\varepsilon \equiv v \)

(iii) \( C(V, \pi, f_1)(y) \in \text{Leaves}(T_2) \)

To establish (a), let \( C(V, \pi, f_{\text{good}})(v) \) be any leaf of \( T_{\text{good}} \) and let \( C(V, \pi, f_1)(y) \) be an arbitrary element of \( \mathcal{M}(C(V, \pi, f_{\text{good}})(v)) \). Then it follows from the definition of the set \( \mathcal{M}(C(V, \pi, f_{\text{good}})(v)) \) that \( C(V, \pi, f_1)(y) \subseteq C(V, \pi, f_1)(z) \) for some spel \( z \) that satisfies the condition \( v \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 2\varepsilon \equiv z \) (which implies \( f_{\text{good}}(z) \geq f_{\text{good}}(v) - 2\varepsilon \)). Since \( C(V, \pi, f_1)(y) \subseteq C(V, \pi, f_1)(z) \), we have that \( z \in (V, \pi, f_1) \geq f(z) \equiv y \). This implies
$z \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(z) - 2\varepsilon \Rightarrow y$ (in view of assertion (ii) of F), which implies $z \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 4\varepsilon \Rightarrow y$ (as $f_{\text{good}}(z) \geq f_{\text{good}}(v) - 2\varepsilon$).

Combining $z \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 4\varepsilon \Rightarrow y$ with $v \in (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 2\varepsilon \Rightarrow z$, we deduce assertion (i) of (a). Now (ii) follows from (i) and D because $C(V, \pi, f_{\text{good}})(v) \in \text{Leaves}(T_{\text{good}})$, and (iii) follows from (ii) and F.

Now we establish (b). Suppose $C(V, \pi, f_{\text{good}})(y) = \phi(C(V, \pi, f_{\text{good}})(v))$. Consider the node $C(V, \pi, f_{\text{good}})(v)$ of $T_1$. Let $S$ be a leaf of $T_1$ such that $S \succeq T_1, C(V, \pi, f_{\text{good}})(v)$. Then we have that $S \in M(C(V, \pi, f_{\text{good}})(v))$, by Lemma B.2. Hence $\ell_1(C(V, \pi, f_{\text{good}})(y)) \geq \ell_1(S)$ (as $S$ cannot occur later in the $\ell_1$-increasing leaf enumeration that is used in pruning $(T_1, \ell_1)$ than $\phi(C(V, \pi, f_{\text{good}})(v)) = C(V, \pi, f_{\text{good}})(y)$, by the definition of $\phi(C(V, \pi, f_{\text{good}})(v))$). Therefore

$$f_1(y) = \ell_1(C(V, \pi, f_{\text{good}})(y)) \geq \ell_1(S) \geq \ell_1(C(V, \pi, f_{\text{good}})(v)) = f_1(v) \quad (B.7)$$

which establishes the second inequality of assertion (i) of (b). The third inequality of (i) follows from F. Now $f_{\text{good}}(v) \geq f_{\text{good}}(y)$ (by assertion (ii) of (a)). This implies $f_{\text{good}}(v) \geq f_1(y) - \varepsilon$ (by F), which is equivalent to the first inequality of assertion (i) of (b). This establishes assertion (i) of (b). It follows from F and assertion (i) of (b) that $f_{\text{good}}(y) \geq f_{\text{good}}(v) - 2\varepsilon$. Assertion (ii) of (b) follows from this and assertion (ii) of (a).

To see that assertion (iii) of (b) holds, let $C(V, \pi, f_{\text{good}})(w)$ be any leaf of $T_1$ that occurs later in the $\ell_1$-increasing leaf enumeration that is used in pruning $(T_1, \ell_1)$ than $\phi(C(V, \pi, f_{\text{good}})(v)) = C(V, \pi, f_{\text{good}})(y)$. Then it follows from the definitions of $\phi(C(V, \pi, f_{\text{good}})(v))$ and of an $\ell_1$-increasing leaf enumeration that:

- $C(V, \pi, f_{\text{good}})(w) \not\in M(C(V, \pi, f_{\text{good}})(v))$
- $f_1(w) = \ell_1(C(V, \pi, f_{\text{good}})(w)) \geq \ell_1(C(V, \pi, f_{\text{good}})(y)) = f_1(y)$

As $f_1(w) \geq f_1(y)$, Equation (B.7) implies that $f_1(w) \geq f_1(v)$, and now it follows
from F that \( f_{\text{good}}(w) \geq f_{\text{good}}(v) - 2\varepsilon \). So \( C_{(V,\pi,f_{\text{good}})}(v) \not\subseteq T_{\text{good}} C_{(V,\pi,f_{\text{good}})}(w) \); otherwise the spell \( w \) would satisfy \( w \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(w) \equiv v \), which would imply that \( w \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(v) - 2\varepsilon \equiv v \) (since \( f_{\text{good}}(w) \geq f_{\text{good}}(v) - 2\varepsilon \)), which would in turn imply that \( C_{(V,\pi,f_1)}(w) \) is an element of \( M(C_{(V,\pi,f_{\text{good}})}(v)) \) (by Lemma B.2), which is false as we saw above.

Since \( C_{(V,\pi,f_{\text{good}})}(v) \not\subseteq T_{\text{good}} C_{(V,\pi,f_{\text{good}})}(w) \), it follows from C and A that \( w \) does not satisfy \( w \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(v) - 4\varepsilon \equiv v \). This and assertion (ii) of F imply that \( w \) does not satisfy \( w \equiv (V,\pi,f_1) \geq f_1(v) - 2\varepsilon \equiv v \), and so (since \( f_1(y) \geq f_1(v) \), by Equation (B.7)) \( w \) does not satisfy \( w \equiv (V,\pi,f_1) \geq f_1(y) - 2\varepsilon \equiv v \). But we know from assertion (iii) of (a) that \( y \equiv (V,\pi,f_1) \geq f_1(y) - 2\varepsilon \equiv v \), so \( w \) also does not satisfy \( w \equiv (V,\pi,f_1) \geq f_1(y) - 2\varepsilon \equiv y \).

As \( C_{(V,\pi,f_1)}(w) \) is an arbitrary leaf of \( T_1 \) that occurs later in the \( \ell_1 \)-increasing leaf enumeration used in pruning \( (T_1,\ell_1) \) than the leaf \( \phi(C_{(V,\pi,f_{\text{good}})}(v)) \), we see from E that \( \phi(C_{(V,\pi,f_{\text{good}})}(v)) \in \text{Leaves}(T_2) \)—i.e., assertion (iii) of (b) holds.

Since \( \phi(C_{(V,\pi,f_{\text{good}})}(v)) \in \text{Leaves}(T_2) \) for every leaf \( C_{(V,\pi,f_{\text{good}})}(v) \) of \( T_{\text{good}} \), we can regard \( \phi \) as a mapping \( \phi : \text{Leaves}(T_{\text{good}}) \to \text{Leaves}(T_2) \), and we will do this from now on.

We next show that \( \phi : \text{Leaves}(T_{\text{good}}) \to \text{Leaves}(T_2) \) is 1-to-1:

H. \( \phi(A) \neq \phi(A') \) whenever \( A \) and \( A' \) are distinct leaves of \( T_{\text{good}} \)

Indeed, let \( C_{(V,\pi,f_{\text{good}})}(v_a) \) and \( C_{(V,\pi,f_{\text{good}})}(v_b) \) be any two distinct leaves of \( T_{\text{good}} \). To establish H, it is enough to show that \( M(C_{(V,\pi,f_{\text{good}})}(v_a)) \) and \( M(C_{(V,\pi,f_{\text{good}})}(v_b)) \) are disjoint. Suppose this is not the case. Then there is a leaf \( C_{(V,\pi,f_1)}(x) \) of \( T_1 \) such that \( C_{(V,\pi,f_1)}(x) \in M(C_{(V,\pi,f_{\text{good}})}(v_a)) \) and \( C_{(V,\pi,f_1)}(x) \in M(C_{(V,\pi,f_{\text{good}})}(v_b)) \).

Now assertion (i) of part (a) of G implies that \( v_a \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(v_a) - 4\varepsilon \equiv x \) and that \( v_b \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(v_b) - 4\varepsilon \equiv x \).

Assuming without loss of generality that \( f_{\text{good}}(v_a) \leq f_{\text{good}}(v_b) \), these two properties imply that \( v_a \equiv (V,\pi,f_{\text{good}}) \geq f_{\text{good}}(v_a) - 4\varepsilon \equiv v_b \), which is impossible in view of C and A. This contradiction establishes H and shows that \( \phi \) is 1-to-1.
Next, we show that:

1. Leaves\((T_2) \setminus \phi[\text{Leaves}(T_{\text{good}})] = \emptyset\)

To justify I, let \( C_{(V,\pi, f_1)}(x) \) be any element of \( \text{Leaves}(T_1) \setminus \phi[\text{Leaves}(T_{\text{good}})] \).

Then what we need to show is that \( C_{(V,\pi, f_1)}(x) \notin \text{Leaves}(T_2) \).

Let \( C_{(V,\pi, f_{\text{good}})}(v) \) be a leaf of \( T_{\text{good}} \) such that \( C_{(V,\pi, f_{\text{good}})}(x) \supseteq C_{(V,\pi, f_{\text{good}})}(v) \).

Then \( x \in (V,\pi, f_{\text{good}}) \supseteq f_{\text{good}}(x) \Rightarrow v \) and so it follows from F that \( x \in (V,\pi, f_1) \geq f_1(x) - 2\varepsilon \Rightarrow v \).

Let \( C_{(V,\pi, f_1)}(y) = \phi(C_{(V,\pi, f_{\text{good}})}(v)) \). We now claim that:

- \( C_{(V,\pi, f_1)}(y) \) occurs later in the \( \ell_1 \)-increasing leaf enumeration that is used in pruning \((T_1, \ell_1)\) than \( C_{(V,\pi, f_1)}(x) \).

Now we justify this claim. Just one of the following is true:

(a) \( f_{\text{good}}(v) - 2\varepsilon > f_{\text{good}}(x) \)

(b) \( f_{\text{good}}(x) \geq f_{\text{good}}(v) - 2\varepsilon \)

In case (a) it follows from F that \( f_1(v) > f_1(x) \), and so \( f_1(y) > f_1(x) \) (since \( f_1(y) \geq f_1(v) \)), by assertion (i) of part (b) of G; thus our claim is valid.

In case (b), we first observe that, since \( x \in (V,\pi, f_{\text{good}}) \supseteq f_{\text{good}}(x) \Rightarrow v \), (b) implies that \( x \in (V,\pi, f_{\text{good}}) \supseteq f_{\text{good}}(v) - 2\varepsilon \Rightarrow v \), so that \( C(x, (V, \pi, f)) \in \mathcal{M}(C(V, \pi, f_{\text{good}})) \) (by Lemma B.2). Therefore \( C_{(V,\pi, f_1)}(x) \in \mathcal{M}(C_{(V,\pi, f_{\text{good}})}(v)) \setminus \{ \phi(C_{(V,\pi, f_{\text{good}})}(v)) \} \), because \( C_{(V,\pi, f_1)}(x) \) is an element of \( \text{Leaves}(T_1) \setminus \phi[\text{Leaves}(T_{\text{good}})] \). As \( C_{(V,\pi, f_1)}(y) = \phi(C_{(V,\pi, f_{\text{good}})}(v)) \) and \( C_{(V,\pi, f_1)}(x) \in \mathcal{M}(C_{(V,\pi, f_{\text{good}})}(v)) \setminus \{ \phi(C_{(V,\pi, f_{\text{good}})}(v)) \} \), it follows from the definition of \( \phi \) that our claim is again valid.

In either case, we have that \( x \in (V,\pi, f_1) \geq f_1(x) - 2\varepsilon \Rightarrow v \) (as we saw above), and the claim implies \( f_1(y) \geq f_1(x) \). So, since we see from assertion (iii) of part (a) of G that \( v \in (V,\pi, f_1) \geq f_1(y) - 2\varepsilon \Rightarrow y \), we also have that \( x \in (V,\pi, f_1) \geq f_1(x) - 2\varepsilon \Rightarrow y \). From this, E, and the above claim, we deduce that \( C_{(V,\pi, f_1)}(x) \notin \text{Leaves}(T_2) \). This justifies I.

It follows from H and I that \( \phi : \text{Leaves}(T_{\text{good}}) \to \text{Leaves}(T_2) \) is a bijection. This completes step 2 of the proof of the Lemma B.1.
B.4 Step 3 of the Proof of the Lemma B.1

We now extend $\phi$ to a mapping $\phi : \text{Crit}(T_{\text{good}}) \rightarrow \text{Crit}(T_2)$ by defining $\phi(U) = \bigwedge_{T_2} \phi [\text{Leaves}(T_{\text{good}}(U))]$. We will establish two properties of the mapping $\phi$ which together imply that $\phi$ is an essential isomorphism of $\mathcal{F}_{\text{good}}$ to $\mathcal{F}_3$. The first property is that, for all $U, U' \in \text{Crit}(T_{\text{good}})$, $\phi(U) \leq_{T_2} \phi(U')$ if, and only if, $U \leq_{T_{\text{good}}} U'$ (so that $\phi$ is an order-preserving injection). The second property is that $\phi[\text{Crit}(T_{\text{good}})] = \text{Crit}(T_3)$. To establish these two properties, we first show that:

J. $|\ell_2(\bigwedge_{T_2} \phi [L]) - \ell_{\text{good}}(\bigwedge_{T_{\text{good}}} L)| \leq \epsilon$ whenever $\emptyset \neq L \subseteq \text{Leaves}(T_{\text{good}})$.

Indeed, suppose $\emptyset \neq L \subseteq \text{Leaves}(T_{\text{good}})$. If $|L| = 1$, then $J$ is an immediate consequence of assertion (i) of part (b) of G, so we will assume $|L| \geq 2$.

For brevity, we will write $\tau_L$ for $\ell_{\text{good}}(\bigwedge_{T_{\text{good}}} L)$ and $\tau_{\phi[L]}$ for $\ell_2(\bigwedge_{T_2} \phi [L])$, so that $J$ can be written as $|\tau_{\phi[L]} - \tau_L| \leq \epsilon$.

We first show that $\tau_{\phi[L]} \geq \tau_L - \epsilon$. For this purpose, let $C_{(V,\pi,f_1)}(x)$ and $C_{(V,\pi,f_1)}(y)$ be any two distinct elements of $\phi[L]$. Then $C_{(V,\pi,f_1)}(x) = \phi(C_{(V,\pi,f_{\text{good}})}(u))$ and $C_{(V,\pi,f_1)}(y) = \phi(C_{(V,\pi,f_{\text{good}})}(v))$, where $C_{(V,\pi,f_{\text{good}})}(u)$ and $C_{(V,\pi,f_{\text{good}})}(v)$ are two distinct elements of $L$. From A and the definition of $\tau_L$ we see that $u \leq (V,\pi,f_{\text{good}}) \geq \tau_L \Rightarrow v$.

This and F imply that $u \leq (V,\pi,f_1) \geq \tau_L - \epsilon \Rightarrow v$. We see from the definition of $\phi$ and assertion (iii) of part (a) of G that $x \leq (V,\pi,f_1) \geq f_1(x) - 2\epsilon \Rightarrow u$ and $y \leq (V,\pi,f_1) \geq f_1(y) - 2\epsilon \Rightarrow v$.

Combining the last three observations, we deduce that:

$$x \leq (V,\pi,f_1) \geq \min(\tau_L - \epsilon, f_1(x) - 2\epsilon, f_1(y) - 2\epsilon) \Rightarrow y$$  \hspace{1cm} (B.8)

However, it follows from C and the definition of $\tau_L$ that

$$\tau_L \leq \ell_{\text{good}}(\bigwedge_{T_{\text{good}}} \{C_{(V,\pi,f_{\text{good}})}(u), C_{(V,\pi,f_{\text{good}})}(v)\})$$

$$< \min(\ell_{\text{good}}(C_{(V,\pi,f_{\text{good}})}(u)) - 4\epsilon, \ell_{\text{good}}(C_{(V,\pi,f_{\text{good}})}(v)) - 4\epsilon)$$

$$= \min(f_{\text{good}}(u) - 4\epsilon, f_{\text{good}}(v) - 4\epsilon)$$
which implies that $\tau_L - \varepsilon < \min(f_{\text{good}}(u) - 5\varepsilon, f_{\text{good}}(v) - 5\varepsilon)$, which implies that $\tau_L - \varepsilon < \min(f_1(u) - 4\varepsilon, f_1(v) - 4\varepsilon)$ (in view of F), which in turn implies that $\tau_L - \varepsilon < \min(f_1(x) - 4\varepsilon, f_1(y) - 4\varepsilon)$ (by assertion (i) of part (b) of G). So Equation (B.8) can be simplified to $x \equiv f_1 \geq \tau_L - \varepsilon$ (since $C(V, \pi, f_1)(x)$ and $C(V, \pi, f_1)(y)$ are arbitrary distinct elements of $\phi[L]$, as required.

To complete the proof of J, we show that $\tau_L \geq \tau_{\phi[L]} - \varepsilon$. This time we let $C(V, \pi, f_{\text{good}})(u)$ and $C(V, \pi, f_{\text{good}})(v)$ be any two distinct elements of $L$, and then define $C(V, \pi, f_1)(x) = \phi(C(V, \pi, f_{\text{good}})(u))$ and $C(V, \pi, f_1)(y) = \phi(C(V, \pi, f_{\text{good}})(v))$, so that $C(V, \pi, f_1)(x), C(V, \pi, f_1)(y) \in \phi[L]$. From A and the definition of $\tau_{\phi[L]}$ we see that $x \equiv (V, \pi, f_1) \geq \tau_{\phi[L]} \equiv y$. This and F imply that $x \equiv (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(u) - 2\varepsilon \equiv x$, we similarly have that $v \equiv (V, \pi, f_{\text{good}}) \geq f_{\text{good}}(v) - 2\varepsilon \equiv y$. Combining the last three observations, we see that:

$$u \equiv (V, \pi, f_{\text{good}}) \geq \min(\tau_{\phi[L]} - \varepsilon, f_{\text{good}}(u) - 2\varepsilon, f_{\text{good}}(v) - 2\varepsilon) \equiv v \quad \text{(B.9)}$$

However, it follows from the definition of $\tau_{\phi[L]}$ and E’ that:

$$\tau_{\phi[L]} \leq \ell_2(\Lambda T_2\{\phi(C(V, \pi, f_{\text{good}})(u)), \phi(C(V, \pi, f_{\text{good}})(v))\})$$

$$\leq \min(\ell_2(\phi(C(V, \pi, f_{\text{good}})(u))), \ell_2(\phi(C(V, \pi, f_{\text{good}})(v)))) - 2\varepsilon$$

$$= \min(\ell_2(C(V, \pi, f_1)(x)), \ell_2(C(V, \pi, f_1)(y))) - 2\varepsilon = \min(f_1(x), f_1(y)) - 2\varepsilon$$

Hence $\tau_{\phi[L]} - \varepsilon < \min(f_1(x) - 3\varepsilon, f_1(y) - 3\varepsilon)$, which (by assertion (i) of part (b) of G) implies $\tau_{\phi[L]} - \varepsilon < \min(f_{\text{good}}(u) - 2\varepsilon, f_{\text{good}}(v) - 2\varepsilon)$. We now see from Equation (B.9) that $u \equiv (V, \pi, f_{\text{good}}) \geq \tau_{\phi[L]} - \varepsilon \equiv v$. It follows from this and A that $\tau_L \geq \tau_{\phi[L]} - \varepsilon$ (since $C(V, \pi, f_{\text{good}})(u)$ and $C(V, \pi, f_{\text{good}})(v)$ are arbitrary distinct elements of $L$), as required. Thus we have established J.

From B and J, we deduce:
K. Whenever $\emptyset \neq L \subseteq L' \subseteq \text{Leaves}(T_{\text{good}})$, $\land_{T_{\text{good}}} \mathbf{L}' = \land_{T_{\text{good}}} \mathbf{L}$ if, and only if, $\ell_2(\land_{T_2} \phi[\mathbf{L}]) - \ell_2(\land_{T_2} \phi[\mathbf{L}']) \leq 2\varepsilon$.

As we show in Appendix C, it is not difficult to deduce from K that:

L. For all $U \in \text{Crit}(T_{\text{good}})$, $\text{Leaves}(T_2[\varphi(U)]) = \varphi[\text{Leaves}(T_{\text{good}}[U])]$.

M. For all $X \in \varphi[\text{Crit}(T_{\text{good}})]$, there is no $Y \in X \downarrow_{T_2} \cap \text{Crit}(T_2)$ that satisfies the condition $\ell_2(X) - \ell_2(Y) \leq 2\varepsilon$.

N. For all $X \in \text{Crit}(T_2)$, some $Z \in X \downarrow_{T_2} \cap \varphi[\text{Crit}(T_{\text{good}})]$ satisfies the condition $\ell_2(X) - \ell_2(Z) \leq 2\varepsilon$.

We mention here that N is proved by showing that for every $X \in \text{Crit}(T_2)$ the node $Z = \varphi(\land_{T_{\text{good}}} \varphi^{-1}[\text{Leaves}(T_2[X])])$ has the stated property.

Using L, it is quite easy to show that:

O. For all $U, U' \in \text{Crit}(T_{\text{good}})$, $\varphi(U) \preceq_{T_2} \varphi(U')$ if, and only if, $U \preceq_{T_{\text{good}}} U'$.

Details of the proof of O are given in Appendix C. It follows from O that $\varphi$ is an order-preserving injection.

As $\mathfrak{F}_3 = (T_3, \ell_3)$ is the result of eliminating internal edges of length $\leq 2\varepsilon$ from $\mathfrak{F}_2^{\text{crit}}$, it follows from M and property E8 of simplification step 3 that $\varphi$ must satisfy $\varphi[\text{Crit}(T_{\text{good}})] \subseteq \text{Crit}(T_2) \cap \text{Nodes}(T_3) = \text{Crit}(T_3)$. Moreover, N implies that, for all $X \in \text{Crit}(T_2) \setminus \varphi[\text{Crit}(T_{\text{good}})]$, some $Z \in X \downarrow_{T_2} \cap \varphi[\text{Crit}(T_{\text{good}})]$ satisfies $\ell_2(X) - \ell_2(Z) \leq 2\varepsilon$. We therefore have that:

- For all $X \in \text{Crit}(T_2) \setminus \varphi[\text{Crit}(T_{\text{good}})]$, some $Z \in X \downarrow_{T_2} \cap \text{Crit}(T_3)$ satisfies the condition $\ell_2(X) - \ell_2(Z) \leq 2\varepsilon$.

From this and property E9 of simplification step 3 we deduce that $\varphi$ satisfies $(\text{Crit}(T_2) \setminus \varphi[\text{Crit}(T_{\text{good}})]) \cap \text{Nodes}(T_3) = \emptyset$. Equivalently, $\varphi$ satisfies the condition $\text{Crit}(T_3) \setminus \varphi[\text{Crit}(T_{\text{good}})] = \emptyset$. Thus $\varphi[\text{Crit}(T_{\text{good}})] = \text{Crit}(T_3)$. So the order-preserving injection $\varphi$ can be regarded as a bijection $\varphi : \text{Crit}(T_{\text{good}}) \rightarrow \text{Crit}(T_3)$. 
When so regarded, \( \varphi \) is an essential isomorphism of \( \mathcal{F}_{\text{good}} \) to \( \mathcal{F}_3 \). Finally, \( \varphi \) is level-preserving to within \( \varepsilon \) because, for any node \( U \in \text{Crit}(T_{\text{good}}) \), we deduce from \( J \) (setting \( L = \text{Leaves}(T_{\text{good}}[U]) \), so that \( \bigwedge_{T_{\text{good}}[U]} L = U \)) that \( |\ell_3(\varphi(U)) - \ell_{\text{good}}(U)| \leq \varepsilon. \)

This completes the proof of the Lemma B.1.
Appendix C

Justification of Assertions L, M, N, and O in Step 3 of the Proof of the Lemma B.1

For any rooted tree $T$ and any $U \in \text{Crit}(T)$, we write $L_T U$ to denote the set $\text{Leaves}(T[U]) = \{ A \in \text{Leaves}(T) \mid U \preceq_T A \}$. It is readily confirmed that the following are true in any rooted tree $T$:

1. If $\emptyset \neq L \subseteq L' \subseteq \text{Leaves}(T)$, then: $\bigwedge_T L' \preceq_T \bigwedge_T L$ (C.1)
2. If $\emptyset \neq L \subseteq \text{Leaves}(T)$, then: $L_T \bigwedge_T L \supseteq L$ (C.2)
3. If $U \in \text{Crit}(T)$, then: $\bigwedge_T L_T U = U$ (C.3)
4. If $U \in \text{Crit}(T)$ and $L \supseteq L_T U$, then: $\bigwedge_T L \preceq_T U = \bigwedge_T L_T U$ (C.4)
5. If $U, A \in \text{Crit}(T)$, then: $L_T A = L_T U$ if and only if $A = U$ (C.5)
6. If $U, A \in \text{Crit}(T)$, then: $L_T A \supseteq L_T U$ if and only if $A \preceq_T U$ (C.6)

For all $L \subseteq \text{Leaves}(T_{\text{good}})$ and all $L' \subseteq \text{Leaves}(T_2)$, we write $\phi L$ to mean $\phi[L]$ and we write $\phi^{-1} L$ to mean $\phi^{-1}[L]$. 
If \( X \preceq_{T_2} Y \) or \( Y \preceq_{T_2} X \), and \( \lambda \) is any positive value, then we write \( X \approx_{\lambda} Y \) to mean that \( |\ell_2(Y) - \ell_2(X)| \leq \lambda \), and write \( X \prec_{\lambda} Y \) to mean that \( \ell_2(Y) - \ell_2(X) > \lambda \); in the latter case we must have that \( X \prec_{T_2} Y \). For brevity, we will write \( \land_{\text{good}} \) and \( \land_2 \) to mean \( \land_{T_{\text{good}}} \) and \( \land_{T_2} \), and write \( \mathcal{L}_{\text{good}} \) and \( \mathcal{L}_2 \) to mean \( \mathcal{L}_{T_{\text{good}}} \) and \( \mathcal{L}_{T_2} \). Note that the definition of the mapping \( \varphi \) can be rewritten in terms of \( \phi \) and \( \mathcal{L}_{\text{good}} \) as follows:

\[
\varphi(U) \overset{\text{def}}{=} \land_2 \phi \mathcal{L}_{\text{good}} U
\] (C.7)

If \( \emptyset \neq L \subseteq L' \subseteq \text{Leaves}(T_{\text{good}}) \), then \( \emptyset \neq \phi L \subseteq \phi L' \subseteq \text{Leaves}(T_2) \) and so \( \land_2 \phi L' \preceq_{T_2} \land_2 \phi L \) (by Equation (C.1)). Hence assertion K can be restated as follows (for all nonempty sets \( L \subseteq L' \subseteq \text{Leaves}(T_{\text{good}}) \)):

\[
\land_2 \phi L' \approx_{2\varepsilon} \land_2 \phi L \text{ if and only if } \land_{\text{good}} L' = \land_{\text{good}} L
\] (C.8)

When \( \emptyset \neq L \subseteq L' \subseteq \text{Leaves}(T_{\text{good}}) \), the negations of \( \land_2 \phi L \approx_{2\varepsilon} \land_2 \phi L' \) and \( \land_{\text{good}} L' = \land_{\text{good}} L \) are \( \land_2 \phi L' \prec_{2\varepsilon} \land_2 \phi L \) and \( \land_{\text{good}} L' \prec_{T_{\text{good}}} \land_{\text{good}} L \) respectively (since \( \land_{\text{good}} L' \preceq_{T_{\text{good}}} \land_{\text{good}} L \) and \( \land_2 \phi L' \preceq_{T_2} \land_2 \phi L \)), so Equation (C.8) can also be stated as follows (for all nonempty sets \( L \subseteq L' \subseteq \text{Leaves}(T_{\text{good}}) \)):

\[
\land_2 \phi L' \prec_{2\varepsilon} \land_2 \phi L \text{ if and only if } \land_{\text{good}} L' \prec_{T_{\text{good}}} \land_{\text{good}} L
\] (C.9)

### C.1 Proof of Assertion L

In view of Equation (C.7), L can be restated as follows:

- For all \( U \in \text{Crit}(T_{\text{good}}) \), we have that \( \mathcal{L}_2 \land_2 \phi \mathcal{L}_{\text{good}} U = \phi \mathcal{L}_{\text{good}} U \). Equivalently, \( \phi^{-1} \mathcal{L}_2 \land_2 \phi \mathcal{L}_{\text{good}} U = \mathcal{L}_{\text{good}} U \).

To prove this, let \( U \in \text{Crit}(T_{\text{good}}) \). Then we successively deduce:
\[ \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \supseteq \phi \mathcal{L}_{\text{good}} U \quad \text{[by (C.2)]} \]
\[ \phi^{-1} \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \supseteq \phi^{-1} \phi \mathcal{L}_{\text{good}} U \]
\[ \phi^{-1} \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \supseteq \mathcal{L}_{\text{good}} U \quad \text{(C.10)} \]

The result will follow from Equation (C.10) if we can show that the following is not true:
\[ \phi \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \supseteq \mathcal{L}_{\text{good}} U \quad \text{(C.11)} \]

To do this, we derive a contradiction from Equation (C.11) as follows:

\[ \bigwedge \text{good} \phi^{-1} \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \prec_{T_{\text{good}}} \bigwedge \text{good} \mathcal{L}_{\text{good}} U \quad \text{[by (C.11) and (C.4)]} \]
\[ \bigwedge \phi^{-1} \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \prec_{2\varepsilon} \bigwedge \phi \mathcal{L}_{\text{good}} U \quad \text{[by (C.9) and (C.10)]} \]
\[ \bigwedge \mathcal{L}_2 \bigwedge \phi \mathcal{L}_{\text{good}} U \prec_{2\varepsilon} \bigwedge \phi \mathcal{L}_{\text{good}} U \]
\[ \bigwedge \phi \mathcal{L}_{\text{good}} U \prec_{2\varepsilon} \bigwedge \phi \mathcal{L}_{\text{good}} U \quad \text{[by (C.3)]} \]

### C.2 Proof of Assertion M

In view of Equation (C.7), M is equivalent to:

- If \( X = \bigwedge \phi \mathcal{L}_{\text{good}} U \) for some \( U \in \text{Crit}(T_{\text{good}}) \), and if \( Y \in \text{Crit}(T_2) \) satisfies \( Y \prec_{T_2} X \), then \( Y \prec_{2\varepsilon} X \).
To prove this, suppose \( X = \bigwedge_2 \phi \mathcal{L}_{\text{good}} U \) for some \( U \in \text{Crit}(T_{\text{good}}) \), and \( Y \in \text{Crit}(T_2) \) satisfies \( Y \preceq_{T_2} X \). Then we can successively deduce:

\[
\begin{align*}
Y & \prec_{T_2} \bigwedge_2 \phi \mathcal{L}_{\text{good}} U & [\text{because } Y \prec_{T_2} X] \\
\mathcal{L}_2 Y & \supseteq \mathcal{L}_2 \bigwedge_2 \phi \mathcal{L}_{\text{good}} U & [\text{by (C.6)}] \\
\mathcal{L}_2 Y & \supseteq \phi \mathcal{L}_{\text{good}} U & [\text{by (C.2)}] \\
\phi^{-1} \mathcal{L}_2 Y & \supseteq \mathcal{L}_{\text{good}} U & (C.12) \\
\bigwedge_{\text{good}} \phi^{-1} \mathcal{L}_2 Y & \prec_{T_{\text{good}}} \bigwedge_{\text{good}} \mathcal{L}_{\text{good}} U & [\text{by (C.4)}] \\
\bigwedge_2 \phi \phi^{-1} \mathcal{L}_2 Y & \prec_{2\varepsilon} \bigwedge_2 \phi \mathcal{L}_{\text{good}} U & [\text{by (C.9) and (C.12)}] \\
\bigwedge_2 \mathcal{L}_2 Y & \prec_{2\varepsilon} \bigwedge_2 \phi \mathcal{L}_{\text{good}} U \\
Y & \prec_{2\varepsilon} \bigwedge_2 \phi \mathcal{L}_{\text{good}} U & [\text{by (C.3)}]
\end{align*}
\]

This proves that \( Y \prec_{2\varepsilon} X \).

### C.3 Proof of Assertion N

In view of Equation (C.7), \( \bigwedge_2 \phi \mathcal{L}_{\text{good}} \bigwedge_{\text{good}} \phi^{-1} \mathcal{L}_2 X \in \varphi[\text{Crit}(T_{\text{good}})] \) for every node \( X \) of \( T_2 \). So N can be proved by establishing that:

- For all \( X \in \text{Crit}(T_2) \), the node \( Z = \bigwedge_2 \phi \mathcal{L}_{\text{good}} \bigwedge_{\text{good}} \phi^{-1} \mathcal{L}_2 X \) satisfies \( Z \preceq_{T_2} X \) and \( X \sim_{2\varepsilon} Z \).
To prove this, let $X \in \text{Crit}(T_2)$ and let $Z = \land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X$. Then we successively deduce:

$$\mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \supseteq \phi^{-1} \mathcal{L}_2 X \quad \text{[by (C.2)]} \quad (C.13)$$

$$\phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \supseteq \phi \phi^{-1} \mathcal{L}_2 X$$

$$\phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \supseteq \mathcal{L}_2 X$$

$$\land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \preceq_{T_2} \land_2 \mathcal{L}_2 X \quad \text{[by (C.1)]}$$

$$\land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \preceq_{T_2} X \quad \text{[by (C.3)]}$$

This proves that $Z \preceq_{T_2} X$. We can also successively deduce:

$$\land_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X = \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \quad \text{[by (C.3)]}$$

$$\land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \approx_{2\varepsilon} \land_2 \phi \phi^{-1} \mathcal{L}_2 X \quad \text{[by (C.8) and (C.13)]}$$

$$\land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \approx_{2\varepsilon} \land_2 \mathcal{L}_2 X$$

$$\land_2 \phi \mathcal{L}_{\text{good}} \land_{\text{good}} \phi^{-1} \mathcal{L}_2 X \approx_{2\varepsilon} X \quad \text{[by (C.3)]}$$

This proves that $Z \approx_{2\varepsilon} X$.

**C.4 Proof of Assertion O**

Let $U, U' \in \text{Crit}(T_{\text{good}})$. Then:

$$U \preceq_{T_{\text{good}}} U' \quad \text{just if} \quad \mathcal{L}_{\text{good}} U \supseteq \mathcal{L}_{\text{good}} U' \quad \text{[by (C.5) and (C.6)]}$$

$$\quad \text{just if} \quad \phi \mathcal{L}_{\text{good}} U \supseteq \phi \mathcal{L}_{\text{good}} U'$$

$$\quad \text{just if} \quad \mathcal{L}_2 \phi(U) \supseteq \mathcal{L}_2 \phi(U') \quad \text{[by assertion L]}$$

$$\quad \text{just if} \quad \phi(U) \preceq_{T_2} \phi(U') \quad \text{[by (C.5) and (C.6)]}$$
Bibliography


## Index

$(\lambda,k)$-simplification, 69  

**A**  

$\alpha$-helix, 116  
ancestor, 26, 43  

**B**  
bacteriophage (cystovirus $\phi 6$), 106  
bandlimit, 3  
band-limited function, 2  
$\beta$-sheet, 116  

**C**  
cardinality, 45  
child, 26, 43  
closest common ancestor, 77  
closest critical descendant, 60  
component inconsistency, 129  
component tree, 44, 62  
component volume, 105  
connectedly embedded digital picture, 38  
convolution, 15  
Coulomb potential, 10  
critical node, 60  
cross-correlation, 20  
cryo-EM, 1, 10  

density, 31  
density map, 4  
descendant, 26, 43  
DIGFile library, 95  
digital picture, 30  
digital picture of an FCTS, 66  
digital space, 30  
docking, 6, 17, 125  

**E**  
edge-adjacency, 33  
edges, 26, 43  
Electron Atomic Scattering Factor, 16  
Electron Microscopy Data Bank, 11  
embedded digital picture, 34, 126  
embedded digital space, 34  
embedding map, 26  
$\varepsilon$-perturbation, 158  
essentially isomorphic, 70  

**F**  
FCTS, 64
FCTS of the digital picture, 66
Figure of Merit (FOM), 136
fitting, 6, 17
flexible docking, 23
foreground component tree structure, 64
foreground history tree, 44

G
graylevel, 31
grid, 4
grid step-size, 4

I
image, 4
intensity, 31
isomorphic digital pictures, 31
isomorphic subtree, 26

J
join tree, 44

L
label, 26
labeled rooted tree, 26
labeled subtree, 130
labeled tree, 129
labeled tree embedding, 129
lattice, 4
leaves, 60, 98
level, 64

level of node, 46
level-preserving essential isomorphism, 78
level-preserving to within δ, 72
lowest critical node, 68
Low-pass EASF, 17

M
magnetic resonance angiograms (MRA), 94
magnetic resonance imaging (MRI), 94
manual docking, 19
MataExplor, 93
maximum intensity extremal regions, 33
minor embedding, 54
model density map, 6
multiple docking, 24

N
natural component embedding, 55
nodes, 26, 43
Nuclear Magnetic Resonance, 10
Nyquist-Shannon sampling theorem, 3

O
ordered labeled rooted tree, 26
Orientation Score, 137

P
parent, 26, 43
partition, 34
PDB file, 13
phenogram-like, 97
π-adjacent, 30
π-component, 31
π-connected, 30
π-path, 30
pixels, 33
procapsid, 107
proper ancestor, 26, 43
proper descendant, 26, 43
Protein Data Bank, 11, 12
protein primary structure, 116
protein quaternary structure, 116
protein secondary structure, 116
protein tertiary structure, 116
proteins, 115

Q
quality-of-fit, 6, 20

R
regularly embedded, 34
resolution, 3, 9
resolution distance, 3
resolution limit, 3
rigid-body docking, 20
root, 26, 43
Root Mean Square Deviation , 136
rooted tree, 26, 43
root-to-root labeled tree embedding, 129

S
sampling frequency, 3
sampling theorem, 3
secondary structure elements (SSE), 116
set of components, 32
single particle reconstruction, 11
SNARK09, 95
spatial frequency radius, 3
spels, 30
step-size, 3
subtree, 80
superlevel set, 31

T
target density map, 6
threshold, 46
Transmission Electron Microscopy, 10
tree embedding, 26, 54, 129
tree embedding problem, 27

U
unordered labeled rooted tree, 26

V
voxel size, 4

X
X-ray crystallography, 1, 9