Statistical Methods for Object Detection in a Three Dimensional Volume

A Thesis Presented

by

Tzipora Halevi

to

Electrical and Computer Engineering

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the field of

Electrical Engineering

Northeastern University

Boston, Massachusetts

January 1998

Abstract

Anomaly detection in a physical medium is a common objective in many applications, including medical imaging, geological exploration and others. The anomaly detection problem is to identify regions of the medium which have significantly different characteristics than the rest of the medium from a given set of measurements.

In this work we develop and implement an algorithm for detection and localization of anomalous objects in a three dimensional volume from noisy data. The algorithm is based on the multi-scale hypothesis testing approach [5]. It starts by considering the whole volume and uses detection and estimation techniques to identify sub-regions where the anomalies are likely to be found. It then continues the search recursively proceeding to finer scale localization. The algorithm includes a mechanism to incorporate into the search prior information about the anomalies (such as the number of anomalies, their sizes and shapes) via a set of penalty functions that are used in the detection procedures.

We also examine feedback methods to improve the outcome of the algorithm: In one method we take a closer look at each anomaly which is found by the algorithm in order to improve the accuracy of its borders, and in the other method we subtract out the effect of the found anomalies and then run the algorithm again in order to find more anomalies.

We present experimental results of computer simulations, testing the algorithm on synthetically generated data. This data consisted of a few different anomaly configurations (including between one to four anomalies) in SNR levels ranging from -30 to $+20$dB. Our tests show that the algorithm achieves very good probabilities of detection and false alarm in SNRs as low as -15dB.

In addition, we present a partial analysis for the complexity of the algorithm. Our partial analysis suggests that the complexity of the algorithm is logarithmic in the size of the medium. This result is also supported by our experimental results.

Acknowledgments

During my master studies in Northeastern University, I have had assistance from many people. First and foremost among them has been Prof. Eric Miller, my thesis advisor. I wish to express my sincere gratitude to Eric for suggesting this area of research and for providing invaluable technical guidance, encouragement and support along the way.

I would like to thank Prof. Anthony Devaney and Prof. Carey Rapport for serving on my thesis committee. I am thankful to many people who influenced my studies and research. A very partial list of people with whom I had the opportunity to work and interact throughout my studies in Northeastern includes Prof. David Brady, Prof. Dana Brooks, Prof. Vinay Ingle, Prof. David Kelly, Prof. Brad Lehman, Prof. Hanoch Lev-Ari and Prof. Ram Raghavan. To my friends and office mates Yael Ehrlich, Roger Doufor, Murat Belge, Adnan Sahin, I wish to thank for their friendship and for many conversations. I am also grateful to Cynthia Bates for helping me with the transfer from part-time to full-time studies.

I would like to thank Dr. Guy Even, Ziv Helman and Dr. Yehudit Hocherman, for working with me in my undergraduate studies, and encouraging me to continue to graduate school. I am happy to acknowledge the generous financial support of the Army Research Laboratory Grant and the Bellamy Memorial Fund, whose assistanship I had the opportunity to get in the first year of my studies.

Special thanks to my mother Edith and my father Mendel, for all their love and support over the years.

And finally, I wish to thank my husband Shai, for encouraging me to go back to school, and for supporting me through all this. Without him, none of this would have been possible. I would also like to thank him for working with me on a part of the complexity analysis.

Contents

Abstract ii

Acknowledgments iv

1 Introduction 1

1.1 Overview . . . 1

1.2 Thesis Contributions . . . 4

1.3 Organization . . . 5

2 Preliminaries 6

2.1 Hypothesis Testing . . . 6

2.1.1 Binary Hypothesis Testing . . . 7

2.1.2 M-ary Hypothesis Testing . . . 9

2.1.3 Hypothesis Testing with Additive Gaussian Noise . . . 10

2.2 Parameter Estimation . . . 11

2.2.1 Amplitude Estimation with Gaussian Noise . . . 11

2.3 Problem Statement . . . 12

2.3.1 Vector Notations . . . 13

2.3.2 Hypothesis Testing . . . 14

3 Algorithm 16

3.1 High Level Description 16
3.2 Localization Tests 19
3.3 Pruning Tests 21
3.4 Penalties 22
3.4.1 Localization Penalties 26
3.4.2 Pruning Penalties 29
3.5 Initial Hypotheses 32
3.6 Feedback 32
3.6.1 Reexamining Found Anomalies 33
3.6.2 Subtracting Data of Found Anomalies 35
3.7 Final Processing 35
3.8 Generalization 37
3.8.1 The Linear Case 37

4 Complexity Analysis **38**
4.1 Hypotheses Testing Complexity 38
4.2 Complexity with no Pruning 39
4.3 Complexity with Pruning 41
4.4 Comparison with Exhaustive Search 43
4.4.1 A Numerical Example 44
4.4.2 Feedback Complexity 45

5 Test Results **46**
5.1 A Single Anomaly 48
5.2 Two Anomalies with the Same Size and Amplitude 51
5.3 Two Anomalies of Different Sizes 55
5.4 Two Anomalies of Different Amplitudes 57
5.5 Experiments with More Anomalies 59

5.6 Complexity 60

6 Conclusions **64**

6.1 Future Work 65

Bibliography **68**

A Penalties are Necessary **70**

A.1 Comparing H_0 and H_1 71

A.2 Comparing H_1 and H_2 73

A.3 More Than one Region 74

A.4 Conclusion 76

B Complexity Analysis with Pruning **77**

List of Figures

2.1 Representing a 3-dimensional array of voxels by a 1-dimensional vector 13

2.2 An illustration of our notations for a 2-dimensional area . . . 14

3.1 A simple example of an execution tree 18

3.2 An overview of the Algorithm 19

3.3 Localization Hypotheses. The dark regions are the ones which are kept in each hypothesis 21

3.4 Pruning hypotheses: an example with four areas. 23

3.5 Localization penalty functions for $MAS = 4$ 26

3.6 The pruning penalty functions: (a) H_0 penalty; (b) H_j penalty. 30

3.7 Initial hypotheses 31

3.8 Example of execution with feedback. The black region is the anomaly and the gray regions are the hypotheses of the algorithm. 34

4.1 An example of an execution tree with many nodes 42

5.1 Test results for a single anomaly 49

5.2 A single anomaly: snapshots of execution 52

5.3 A single anomaly: snapshots of feedback 53

5.4 Two anomalies with the same size and amplitude 54

5.5 Two anomalies of different sizes 56

5.6 Two anomalies of different amplitudes 58

5.7 Example with four anomalies 60

5.8 Example with 3 anomalies 61

5.9 Hypothesis-testing complexity vs. region-size 62

A.1 Localization hypotheses H_0, H_1 and H_2 72

List of Tables

Chapter 1

Introduction

1.1 Overview

The problem of *anomaly detection* has a wide and diverse range of manifestations. A few examples include detecting the existence of tumors in medical imaging [7], detecting oil bearing regions from electrical conductivity measurements in the earth [11], detecting mines in an image obtained from a ground penetrating radar [3], detecting "red eyes" in a family picture, and perhaps even detecting an unusual behavior in the stock market.

In this thesis we study an algorithm for anomaly detection in a three-dimensional physical medium from noisy data. Our goal is to identify regions in the medium which are quite different than the rest of the medium. A little more precisely, we assume that we have noisy measurements of some physical quantity (e.g., electrical permitivity) in a volume, and we try to identify regions where this quantity differs in a statistically significant manner from the rest of the medium (below we call this physical quantity the *intensity* of the region).

A natural approach towards this detection problem is to use statistically based methods. These methods provide a mathematical model in which we can describe the medium, analyze the deterministic and stochastic properties of the data and estimate measurement errors. Using this model provides us with insights into the problem at hand, and allows us to use some of the well developed techniques for detection and estimation to obtain an efficient algorithm for solving the problem.

In the last two decades there has been a large body of work regarding the anomaly detection problem in situations where the available data is obtained by measurements along the exterior of the medium, and is therefore a function (e.g., projection) of the actual intensity field of the medium.

One approach for solving this problem is to first reconstruct an image of the intensity field from the measurement and then to process this image in order to detect the anomalies. The problem of reconstructing the intensity field from the measured data (known as the *inverse problem*) has been significantly researched and several standard general purpose methods have been developed for its solution. Among these methods are the filtered back-projection(FBP) and convolution back-projection (CBP) algorithms for tomographic reconstruction problems [1]. These techniques, however, typically rely on low noise data and on the availability of a full set of data. In high levels of noise or sparse data, the problem becomes highly ill-posed, thus necessitating the use of regularization methods. These methods, however, typically result in a blurred reconstructed image in locations where the anomalies are supposed to be, hence rendering the anomaly-detection methods less effective.

Another approach to detection is characterization of anomalies directly from the measured data, without reconstruction of the full image. In [9], Rossi and Willsky consider the problem of characterizing a set of parameters directly from limited, noisy measured CT data. Specifically, they develop a method for estimating the location of a single object with known shape and density situated within a known background field, where the measured data consists of noisy projections of the intensity.

In [5], Miller and Willsky introduce a *scale recursive approach* for solving the anomaly-detection problem in a two-dimensional medium in the context of the linearized scattering problem. They develop an algorithm for detecting multiple anomalies of a more general class of objects than that considered by Rossi and Willsky. In this setting, the observed data is a linear transformation of the intensity field with additive noise. On a high level, the Miller-Willsky approach starts from coarse-scale/low-resolution localization,and gradually progresses to finer-scale/higher-resolution regions, until the actual anomalies are found. Specifically, the algorithm repeatedly partitions the area at hand, zooming into regions where anomalies are "more likely" to exist and proceeds recursively to search for anomalies in these regions. In each step along the way, methods of hypothesis testing and parameter estimation are employed to decide on the next region to zoom in. A later work by Miller [4] extends this approach for a non-linear scattering model. Miller also introduces a method for inclusion of prior information to guide the search, via a set of penalty terms that play the role of prior probabilities in the hypothesis-testing procedures. These penalty terms are used to represent prior expectations about the number of anomalies, their sizes and shapes. In [2], Frakt considers the use of multi-scale hypothesis testing which do not necessarily correspond to spatial zooming, in the context of characterization of a single anomaly from tomographic projections. He considered the cases where the background (i.e., normal activity of the medium) is either white or fractal, demonstrates that spatial zooming is adequate for the case of white background but not for fractal background, and examines different statistics which can be used for the later case.

In [8], Riley and Devaney develop a method for object localization from two dimensional image data using *wavelet transform* properties. It is shown in this work that the wavelet transforms of an image naturally correspond to multi-scale analysis of this image, and thus can be used to carry out a spatial zooming procedure in a computationally efficient manner. They demonstrate that this approach is more efficient than the standard energy detector, and in fact results in a better probability of detection.

1.2 Thesis Contributions

In this work we design an algorithm which extends the approach of [5, 4] in a few aspects, implement this algorithm and evaluate its performance. We stress that the algorithm in this thesis was developed assuming that we already have a full image of the medium with no distortion. That is, we assume that the measured data on which we apply the algorithm is obtained from the intensity field simply by adding white Gaussian noise. This simple model enables us to analyze different aspects of the algorithm.

Since the Miller-Willsky approach is designed to also handle the case where the fully reconstructed image is not available, our work can be thought of as first step towards realizing an algorithm that works with or without full reconstruction. In Section 3.8 we describe the modifications which must be made in the algorithm in order to generalize it to the no-reconstruction case. The main contributions of this thesis are the following:

- The algorithms in [5, 4] are developed for a two-dimensional medium. In this work we generalize them for three-dimensional medium.

- We investigate the use of feedback techniques in order to improve the localization results. That is, after running the algorithm the results were fed back to the algorithm in order to improve its accuracy.

Specifically we tested two feedback methods. The first method consists of a closer inspection of each anomaly which is found by the algorithm in order to improve the accuracy of its borders. In the second method, we try to find anomalies that might have been missed by the algorithm by subtracting out the effect of the found anomalies and then running the algorithm again.

- We implemented the algorithm and tested it in several different settings to evaluate the effect of the various parameters involved.

1.3 Organization

This thesis is organized as follows: In Chapter 2 we provide a brief review of detection and estimation techniques and describe the formal model in which our algorithm works. Chapter 3 describes the algorithm itself, the various parameters involved, and our implementation. Chapter 4 contains complexity analysis of the algorithm, Chapter 5 describes the experimental results which we obtained, and in Chapter 6 we present our conclusions and possible future research directions.

Chapter 2

Preliminaries

Our anomaly detection algorithm is based on statistical methods of detection and estimation. In particular, we use techniques for M-ary hypothesis testing and least-squared-error estimation. A comprehensive treatment of these and other methods in the theory of detection and estimation can be found in many text books (e.g, [12]). Below we only describe the parts of this theory that are directly relevant for our algorithm.

2.1 Hypothesis Testing

Many signal detection problems can be cast in the framework of M-ary hypothesis testing. In this framework, we have a number of possible statistical explanations (called *hypotheses*) for describing a given stochastic process. Given the outcome of that process (which in our case will be a vector of real numbers), our goal is to pick the hypothesis which is most likely to have generated that outcome. Thus we need to devise a *decision rule* which maps any observed outcome to the best hypothesis describing that outcome.

2.1.1 Binary Hypothesis Testing

For simplicity of notations, throughout this chapter we only discuss hypothesis testing where the observed data is a vector of real numbers. Although generalization to other settings is not hard, we do not use it in the thesis.

We start from the simplest case of hypothesis testing in which we have only two hypotheses. This is called the binary hypothesis testing (BHT) problem (or the detection problem). Traditionally, the two hypotheses are denoted by H_0 and H_1, where H_0 is sometime called the *null hypothesis*. Formally, a BHT problem is specified by two probability density functions, $P(\mathbf{y}; H_0)$ and $P(\mathbf{y}; H_1)$, which define two probability distributions over real vectors in $\mathcal{R}^N$. The interpretation is that $P(\mathbf{y}; H_i)$ is the distribution of the outcome if hypothesis H_i is true. In addition, we sometimes also have some other knowledge about the problem, such as a-priori probabilities of the two hypotheses H_0 and H_1.

Decision Rules

A decision rule for a BHT problem is a mapping $h : \mathcal{R}^N \to \{0,1\}$. The interpretation is that for an observed data $\mathbf{y} \in \mathcal{R}^N$, $h(\mathbf{y})$ is either H_0 or H_1. Two decision rules that are often used in detection theory are the *maximum likelihood* rule (ML) and the *maximum a-posteriori probability* rule (MAP).

The *likelihood* of a hypothesis H_i given the observed data $\mathbf{y}$ is just the probability density of Y under H_i, namely $P(\mathbf{y}; H_i)$. The ML decision rule tells us to simply pick the hypothesis with the maximum likelihood. That is, we have

$$h_{\mathrm{ML}}(\mathbf{y}) = \begin{cases} 0 & \text{if } P(\mathbf{y}; H_0) \geq P(\mathbf{y}; H_1) \\ 1 & \text{otherwise} \end{cases}$$

In settings where we have a-priory probabilities of H_0 and H_1, (which we denoted here by p_0, p_1, respectively), the a-posteriori probability of hypothesis H_i with the observed data $\mathbf{y}$ is defined as $p_i \cdot P(\mathbf{y}; H_i)$. The MAP decision rule is then defined by

$$h_{\mathrm{MAP}}(\mathbf{y}) = \begin{cases} 0 & \text{if } p_0 \cdot P(\mathbf{y}; H_0) \geq p_1 \cdot P(\mathbf{y}; H_1) \\ 1 & \text{otherwise} \end{cases}$$

The likelihood ratio test. We note that both ML and MAP are in fact special cases of the *likelihood ratio test* (LRT). The likelihood ratio of an observed data $\mathbf{y}$ is defined as

$$L(\mathbf{y}) \overset{\mathrm{def}}{=} \frac{P(\mathbf{y}; H_1)}{P(\mathbf{y}; H_0)}$$

For any positive real number γ, the LRT rule with threshold γ is then defined

$$h_{\mathrm{LRT}}(\mathbf{y}) = \begin{cases} 0 & \text{if } L(\mathbf{y}) \leq \gamma \\ 1 & \text{if } L(\mathbf{y}) > \gamma \end{cases}$$

It is easy to see that the ML rule is obtained by setting $\gamma = 1$, and the MAP rule is obtained by setting $\gamma = p_0/p_1$.

Performance Measurements

A useful way to measure the performance of a given decision rule is provided by measuring the *probability of detection* P_d, and the *probability of false-alarm* P_f. For a given BHT problem and a given decision rule h, these are defined by

$$\begin{aligned} P_d &\overset{\mathrm{def}}{=} \Pr[h(\mathbf{y}) = 1 \text{ assuming } H_1 \text{ is true}] \\ P_f &\overset{\mathrm{def}}{=} \Pr[h(\mathbf{y}) = 1 \text{ assuming } H_0 \text{ is true}] \end{aligned}$$

It is clear that to get a "good decision rule" we need to maximize P_d and at the same time to minimize P_f. It is just as clear, however, that these are conflicting objectives. Thus, in some settings we have an upper bound

α on P_f, and we try to maximize P_d subject to the constraint that $P_f \leq \alpha$. The importance of the likelihood-ratio test in this context is demonstrated by the Neyman-Pearson theorem, which asserts that for any α, the decision rule which maximizes P_d subject to the constraint $P_f \leq \alpha$ is an LRT rule. More precisely, we have

Theorem [Neyman-Pearson [6]]: Let H_0, H_1 be a BHT problem, and let α be any real number $\alpha \in (0,1)$. Then there exists a threshold $\gamma > 0$, so that the LRT rule with threshold γ achieves the maximum P_d among all the decision rules for which $P_f \leq \alpha$.

2.1.2 M-ary Hypothesis Testing

Much of the theory of binary hypothesis testing can be extended for any number of hypotheses $M \geq 2$. An M-ary hypothesis testing (MHT) problem is specified by M probability density functions $P(\mathbf{y}; H_i)$ $(i = 0, 1, \ldots, M-1)$, each defining a probability distribution over vectors in $\mathcal{R}^N$. Also, in some settings we have the prior probabilities for the hypotheses which are denoted $(p_0, p_1, \ldots, p_{M-1})$. A decision rule for an MHT problem is a mapping $h : \mathcal{R}^N \to \{0, 1, \ldots, M-1\}$.

As for the binary case, the ML and MAP decision rules are often used in M-ary hypothesis testing. The ML rule is defined by

$$h_{ML}(\mathbf{y}) = \mathrm{argmax}_j \; P(\mathbf{y}; H_j)$$

and in cases where we have priors, the MAP rule is defined by

$$h_{MAP}(\mathbf{y}) = \mathrm{argmax}_j \; p_j \cdot P(\mathbf{y}; H_j)$$

2.1.3 Hypothesis Testing with Additive Gaussian Noise

In the thesis we use the hypothesis testing techniques from above in a setting where the observed data is obtained by adding a Gaussian random vector (called *the noise*) to some fixed vector in $\mathcal{R}^N$. More precisely, let $\mathbf{g}_1, \ldots, \mathbf{g}_{M-1}$ be vectors in $\mathcal{R}^N$, and let $\mathbf{n}$ be an N-dimensional Gaussian vector, $\mathbf{n} \sim \mathcal{N}(\mu, \mathbf{R})$ (where μ is the mean vector and $\mathbf{R}$ is the covariance matrix). Then we define the MHT problem as follows:

$$\begin{aligned} H_0 &: \mathbf{y} = \mathbf{n} \\ H_i &: \mathbf{y} = \mathbf{g}_i + \mathbf{n} \ (i = 1, \ldots, M-1) \end{aligned}$$

If we have prior probabilities $p_0, p_1, \ldots, p_{M-1}$ for $H_0, H_1, \ldots, H_{M-1}$, respectively, then the MAP rule for this problem is

$$h_{MAP}(\mathbf{y}) = \text{argmax}_j \ p_j \cdot \frac{1}{\sqrt{2\pi N}} \exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{g}_j - \mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{g}_j - \mu)\right)$$

where we define $\mathbf{g}_0 = \mathbf{0}$. Since all we care about in the above expression is the index j which maximizes the right-hand-side, we can get rid of the constant $\frac{1}{\sqrt{2\pi N}}$and take the log of the rest, to obtain

$$h_{MAP}(\mathbf{y}) = \text{argmax}_j \left(\log p_j - \frac{1}{2}(\mathbf{y} - \mathbf{g}_j - \mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{g}_j - \mu)\right)$$

Finally, if we define $\pi_j \overset{\text{def}}{=} -2 \log p_j$, then the above rule is equivalent to

$$h_{MAP}(\mathbf{y}) = \text{argmin}_j \left((\mathbf{y} - \mathbf{g}_j - \mu)^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{g}_j - \mu) + \pi_j\right)$$

In particular, in the thesis we use the above in the case where the noise is white and has zero mean (namely, $\mathbf{R} = \sigma^2 I$ and $\mu = \mathbf{0}$). In this case, the last expression is simplified to

$$h_{MAP}(\mathbf{y}) = \text{argmin}_j \left(\frac{(\mathbf{y} - \mathbf{g}_j)^T (\mathbf{y} - \mathbf{g}_t)}{\sigma^2} + \pi_j\right) \tag{2.1}$$

2.2 Parameter Estimation

An *estimation problem* consists of observing a stochastic process whose overall nature is known, but for which some of the parameters are unknown, and trying to estimate those parameters from the outcome of this process. In this thesis, the outcome of the process is always a vector of real numbers, and the unknown parameter is also a vector of real numbers.

Formally, such an estimation problem is specified by a family of probability density functions $\{P(\mathbf{y}; a) \ : \ a \in \mathcal{R}\}$, with the interpretation that $P(\mathbf{y}; a)$ is the distribution over the outcome $\mathbf{y}$ if the unknown parameter is equal to a. An *estimator* for such a problem is a function $\hat{a} : \mathcal{R}^N \rightarrow \mathcal{R}$, which maps each possible outcome $\mathbf{y} \in \mathcal{R}^N$ to an estimation $\hat{a}(\mathbf{y})$ for the unknown parameter a. A very useful estimator in many cases (including the ones in this work) is the *maximum-likelihood estimator*, which is defined

$$\hat{a}_{ML}(\mathbf{y}) \stackrel{\text{def}}{=} \text{argmax}_a P(\mathbf{y}; a)$$

2.2.1 Amplitude Estimation with Gaussian Noise

In the thesis we use maximum-likelihood estimation in order to estimate the amplitude of the anomalies in the presence of Gaussian noise. In this setting we have the observation

$$\mathbf{y} = \mathbf{B}\mathbf{a} + \mathbf{n}$$

where $\mathbf{B}$ is a known matrix, $\mathbf{n}$ is a zero mean Gaussian random vector $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, and $\mathbf{a}$ is a vector of unknown real numbers which we try to estimate. It is easy to show that in this case, the maximum-likelihood estimation is

$$\hat{\mathbf{a}}(\mathbf{y}) = \text{argmin}_{\mathbf{a}}[(\mathbf{y} - \mathbf{B}\mathbf{a})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{B}\mathbf{a})]$$

To solve this, we differentiate with respect to $\mathbf{a}$ and get

$$(\mathbf{Ba})^T\mathbf{R}^{-1}\mathbf{B} - \mathbf{y}^T\mathbf{R}^{-1}\mathbf{B} = 0$$

Solving the last equation yields the estimate

$$\hat{\mathbf{a}}(\mathbf{y}) = \left(\mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\right)^{-1} \cdot \mathbf{B}^T\mathbf{R}^{-1}\mathbf{y} \tag{2.2}$$

2.3 Problem Statement

Next we describe the formal model of the anomaly detection problem which we study in this thesis. On a high level, our goal is to detect and characterize rectangular anomalous regions in a rectangular 3-dimensional volume. We view the volume as an array of voxels, where each voxel represents a cubic volume cell. We assume that voxels have a nominal value of some uniform intensity. We represent the data by a vector of voxel intensities, which is obtained by ordering the 3-dimensional volume into a 1-dimensional vector in a row-column-layer order. An example of this ordering is depicted in Fig. 2.1.

We assume that the medium has some normal intensity, and that the anomalous regions have intensities which are different than the normal one. For simplicity the normal intensity is taken to be zero. Furthermore, the intensity within each anomaly is assumed to be uniform. The measured data is given by adding white Gaussian noise to the voxel intensities. We number the voxels consecutively from 1 to v (the total volume), and denote the vector of voxel intensities by $\mathbf{g}$, the vector of measured intensities by $\mathbf{y}$, and the noise vector by $\mathbf{n}$. In this notation, the i'th entry in $\mathbf{g}$ is the intensity of voxel i, which is zero if this voxel is not contained in any anomaly, and is equal to the intensity of the anomaly which contains voxel i otherwise. Similarly, the i'th

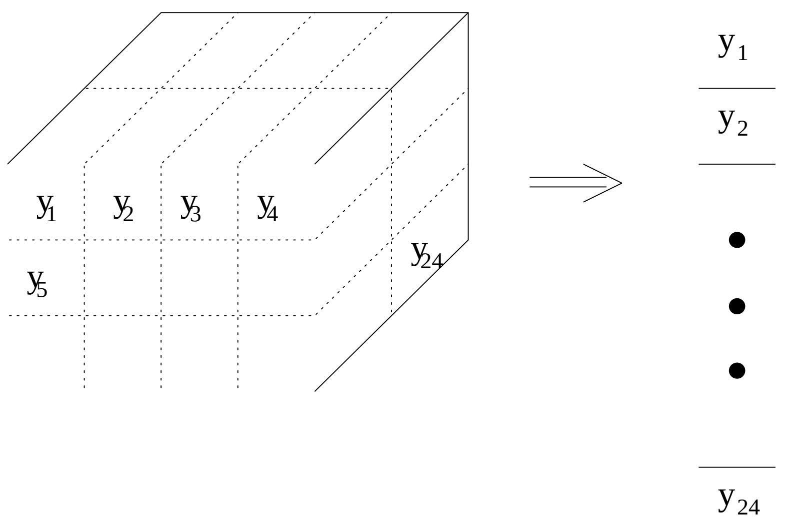


Figure 2.1: Representing a 3-dimensional array of voxels by a 1-dimensional vector

entries in $\mathbf{n}$ and $\mathbf{y}$, respectively, are the noise and measured intensities for the i'th voxel. Using this notation we can write $\mathbf{y} = \mathbf{g} + \mathbf{n}$, where $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$.

2.3.1 Vector Notations

We denote the total number of anomalies by N, and we number them consecutively from 1 to N. For each anomaly j we define an indicator vector $\mathbf{b}_j \in \{0,1\}^v$, where $\mathbf{b}_j[i] = 1$ if voxel i belongs to anomaly j and 0 otherwise. Also, we denote the intensity of anomaly j by a_j.

We now consider the $v \times N$ matrix $\mathbf{B}$, whose j'th column is $\mathbf{b}_j$, and the $N \times 1$ vector $\mathbf{a}$ whose j'th entry is a_j. Using this notation we can write the vector of voxel intensities as $\mathbf{g} = \sum_j a_j \mathbf{b}_j = \mathbf{Ba}$. Thus we can write

$$\mathbf{y} = \mathbf{Ba} + \mathbf{n}$$

Our goal, given the vector $\mathbf{y}$, is to detect the number, locations and sizes of the anomalies, and to estimate their intensities. Using the above notations,

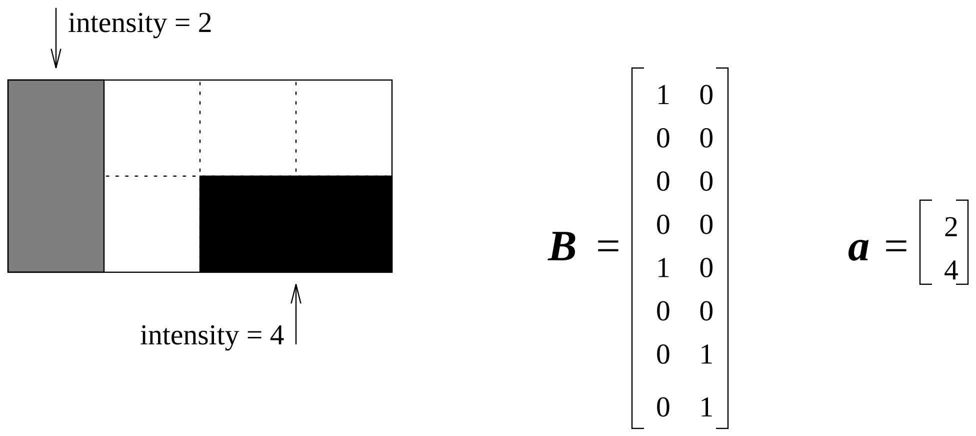

intensity = 2
intensity = 4
$\boldsymbol{B} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$
$\boldsymbol{a} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

Figure 2.2: An illustration of our notations for a 2-dimensional area

this means to find N, and to estimate the 0-1 matrix $\mathbf{B}$ and the real vector $\mathbf{a}$. See Fig. 2.2 for illustration of these notations for a simple 2-dimensional case.

2.3.2 Hypothesis Testing

Using the notations above, the goal of our algorithms can be described as estimating the matrix $\mathbf{B}$ and the vector $\mathbf{a}$ from measured vector $\mathbf{y}$. This is done via a sequence of hypothesis tests. In each step, the algorithm considers a few hypotheses $H_0, H_1, \ldots$, which are formally described using the same notations as above. That is, each hypothesis H_j is described by a matrix $\mathbf{B}_j$ and an estimated-vector $\hat{\mathbf{a}}_j$, so that we have

$$H_j : \mathbf{y} = \mathbf{B}_j \hat{\mathbf{a}}_j + \mathbf{n} \tag{2.3}$$

In each step, the algorithm computes the matrices $\mathbf{B}_j$ from a small fixed set of possibilities (See Section 3.2 for details), and then uses the maximum-likelihood estimator from Eq. 2.2 to compute the vectors $\hat{\mathbf{a}}_j$. Namely, it sets

$$\hat{\mathbf{a}}_j = \left(\mathbf{B}_j^T \mathbf{R}^{-1} \mathbf{B}_j\right)^{-1} \cdot \mathbf{B}_j^T \mathbf{R}^{-1} \mathbf{y} \tag{2.4}$$

Once all the $\mathbf{B}_j$'s and $\hat{\mathbf{a}}_j$'s are set, the algorithm uses MAP procedure to choose among these hypotheses, by choosing the hypothesis H_m such that

$$m = \text{argmin}_j \ \frac{1}{2}(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j)^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j) + \pi_j \tag{2.5}$$

where $\pi_j = -\frac{1}{2}\log p_j$, and p_j is the prior probability of hypothesis H_j.

Chapter 3

Algorithm

3.1 High Level Description

In this chapter we present our algorithm for detection and characterization of an unknown number of anomalous areas in a three-dimensional volume. This algorithm is scale-recursive, and it is based on a sequence of MAP hypothesis tests, of the type discussed in Section 2.3.2. Our general approach is "partition and zoom". That is, we start from a large area and use M-ary hypothesis testing techniques to repeatedly "zoom in" on smaller areas which are likely to contain anomalies. In this process we use two types of tests. One is *localization test* which "zooms in" on the true anomalies, and the other is *pruning test*, which is designed to eliminate falsely identified regions thereby controlling the amount of work required to identify true anomalous structures. We repeat this sequence of localization and pruning tests until we decide that no further localization or pruning is necessary. At this point we conclude that we found the true anomalies.

During the execution of the algorithm we maintain a list of *current regions*. These are the regions where we expect the anomalies to exist. This list is initialized to have one region - the whole area under consideration. In each stage of the algorithm we apply two transformations to the current list. First we perform a single *localization* step, in which we pick one of the regions in the list and sub-divide it according to one of the hypotheses above. Then we perform one or more *pruning* tests, in which we may remove from the list regions that are likely to be "false alarms". This procedure is terminated when none of these transformations changes the current list.

A useful way to think about this procedure is to consider the "execution tree" which consists of all the regions that were on the current list at any time during a particular execution. At the root of this tree we have the whole volume, and we say that region A is a parent of region B in the tree if B was placed on the current list by splitting A during one of the localization steps. Each step in the execution can therefore be viewed as picking one of the leaves in the tree and either splitting it or dropping it altogether. A toy example of such an execution tree (for a 2D volume) is depicted in Fig. 3.1.

After running the localizing and pruning procedure, we add a feedback procedure to improve the accuracy of the algorithm. This procedure re-examines each of the found anomalies separately, by re-running the algorithm only on the close neighborhood of that anomaly. To guarantee a small false-alarm probability, we then perform final processing in which we examine each found anomaly and compute the probability of it being a false alarm. If this probability exceeds some specified threshold, we drop the anomaly from the final configuration.

At this point we might conclude the search, or use a second method of feedback, which consists of subtracting the data associated with the found anomalies from the original data, and repeating the entire process again on the new data. This type of feedback allows us to find additional anomalies

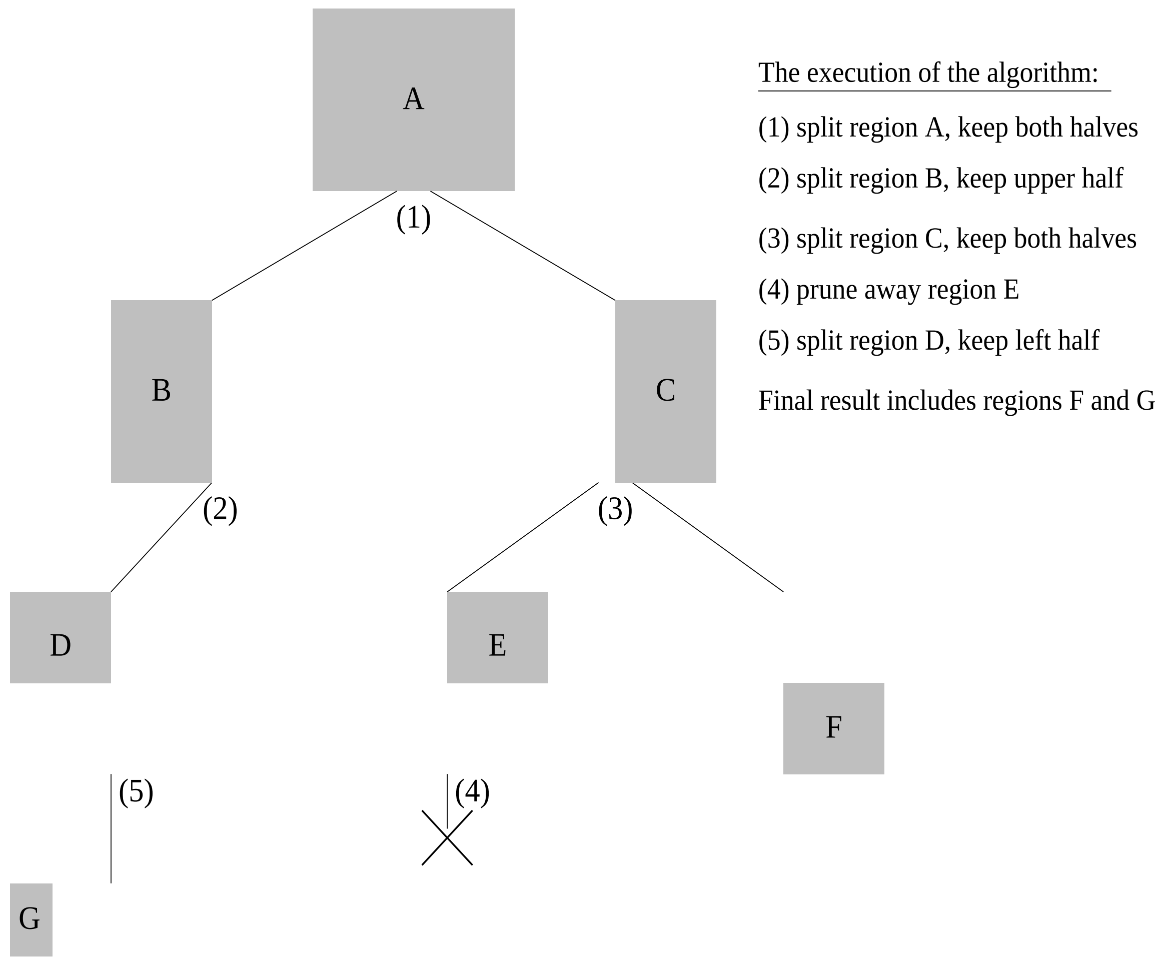

A
B
C
D
E
F
G
(1)
(2)
(3)
(4)
(5)
The execution of the algorithm:
(1) split region A, keep both halves
(2) split region B, keep upper half
(3) split region C, keep both halves
(4) prune away region E
(5) split region D, keep left half
Final result includes regions F and G

Figure 3.1: A simple example of an execution tree

```
 1. Start with one rectangle, which corresponds to the entire volume
 2. Repeat until no change occurs
    2a. Apply M-ary hypothesis testing to choose from among a few
        fixed ways to subdivide one of the current regions
    2b. Apply M-ary hypothesis test in order to prune regions
        which are not likely to contain anomalies
    2c. Repeat Step 2b until no more regions are pruned
3. Apply feedback methods to improve accuracy
4. Remove regions with high probability of false-alarm
```

Figure 3.2: An overview of the Algorithm

that might have been missed by the algorithm. This may be useful in case some anomalies are much smaller in magnitude then others, or in case there are many anomalies and can not all be found at once.

The rest of this chapter is organized as follows: Sections 3.2 and 3.3 describe in details the localization and pruning procedures, and Section 3.4 presents the mechanism of *penalty functions* by which we incorporate prior expectations to the search. Section 3.5 discusses some initialization issues, and Sections 3.6 and 3.7 describe the feedback methods and the final processing stage. Finally, Section 3.8 discusses a few possible generalizations of this work.

3.2 Localization Tests

In the localization steps, we apply a (small) fixed set of hypothesis-tests to each region in the current list (i.e., each column in the current $\mathbf{B}$ matrix), to decide which region (if any) should be subdivided next. The hypothesis test can indicate one of the following:

- No further localization is necessary. This means that no region should be sub-divided, and in fact means that the current configuration is considered the "real" anomaly structure.
- Further refinement is needed. In this case, exactly one of the regions in the current structure is further subdivided.

For simplicity we restrict our algorithm to work only with rectangular areas (though in general it is possible to choose different shapes for this algorithm). For each region in the current list (which is associated with a column in the current matrix $\mathbf{B}$) we examine 12 hypotheses, corresponding to 12 different sub-division configurations. These sub-divisions are depicted in Fig. 3.3. As can be seen in the figure, hypotheses H_2–H_4, H_6–H_8, and H_{10}–H_{12} correspond to sub-divisions which keep exactly one half (either left-half, middle-half or right-half) of the examined region in each dimension. Since it is possible that the current region contains multiple anomalies, then hypotheses H_1, H_5 and H_9 correspond to sub-divisions which keep both the left and the right half in each dimension.

Formally, assume that the current list consists of M regions. Then, our current description of the data is $\mathbf{y} = \mathbf{B}\hat{\mathbf{a}} + \mathbf{n}$, where $\mathbf{B}$ has M columns, $\mathbf{b}_1, \ldots, \mathbf{b}_M$, corresponding to the M regions on the list. In this case, we have $12M + 1$ localization hypotheses to consider.

$$H_j : \mathbf{y} = \mathbf{B}_j\hat{\mathbf{a}}_j + \mathbf{n}$$

Hypothesis H_0 is the null hypothesis which corresponds to retaining the current structure, and therefore we have $\mathbf{B}_0 = \mathbf{B}$ and $\mathbf{a}_0 = \mathbf{a}$. Each of the other hypotheses correspond to dividing exactly one of the M regions according to one of the hypotheses depicted in Fig. 3.3. Specifically, if column $\mathbf{b}_i$ in the

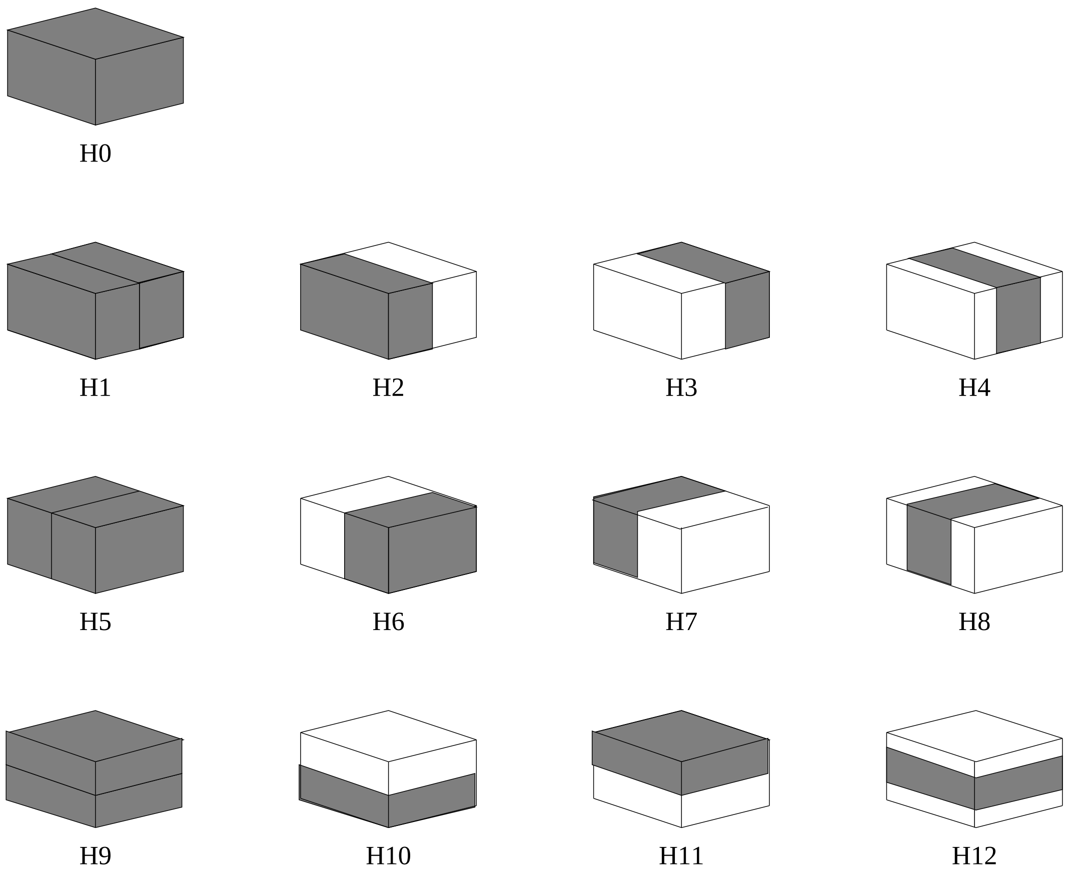


Figure 3.3: Localization Hypotheses. The dark regions are the ones which are kept in each hypothesis

$\mathbf{B}$ matrix corresponds to the i'th region on the list, then each of the matrices $\mathbf{B}_j,\ j = 12i - 11, \ldots 12i$ is obtained by replacing the column $\mathbf{b}_i$ in $\mathbf{B}$ with one or more columns corresponding to sub-divisions of the i'th region according to one of the twelve hypotheses. The vector $\mathbf{a}_j$ is then computed as in Eq. 2.4.

3.3 Pruning Tests

Pruning is designed to eliminate from consideration previously identified regions which we believe are unlikely to actually contain anomalies. This way, we control the amount of work required to identify true anomalous structures by avoiding further sub-divisions of these "unlikely regions" and reduce false alarms. Recall that at each localization step, the algorithm sub-divides a single region. It may be, therefore, that regions which were localized a few iterations ago become less likely than the new acquired regions. The decision of which regions should be eliminated is dictated by our prior assumptions concerning the number and sizes of the anomalies.

To determine which regions should be eliminated (if any), we use a different set of hypothesis tests. The first hypothesis H_0 corresponds to keeping all the regions, and any other hypothesis H_i corresponds to removing region i and keeping the rest of the regions. Formally, hypothesis H_0 is described by keeping the same $\mathbf{B}$ matrix, and each other hypothesis H_i is described by a matrix $\mathbf{B}_i$ which is obtained from $\mathbf{B}$ by removing the i'th column. An example of these hypotheses is depicted in Fig. 3.4.

After each localization step, we run the pruning tests repeatedly until no regions are removed. Specifically, suppose that we have n regions after the localization and before the pruning test. If the test shows that we should not prune any region (namely, if we pick hypothesis H_0), then we are done with the pruning. Else, if we remove region i (namely if we pick hypothesis H_i), then we are left with $n-1$ regions. We now run the pruning test sequence on the remaining $n-1$ regions. This process is repeated until no further regions are removed.

3.4 Penalties

A common problem with search algorithms such as above, is that they tend to over-fit the data. Namely, the algorithm tends to find many small anomalies

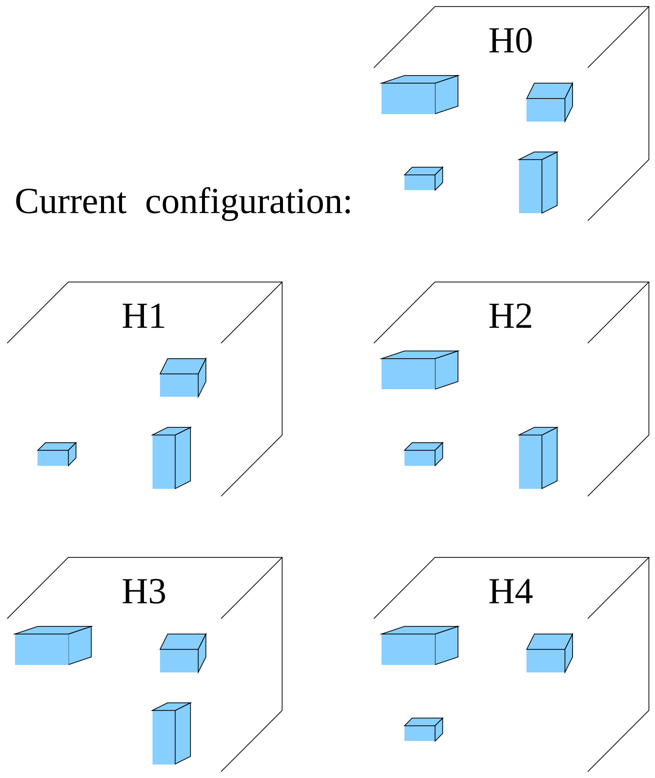


Figure 3.4: Pruning hypotheses: an example with four areas.

instead of a single large one, simply because the data measured at different parts of the anomaly varies slightly due to the noise.

We analyze this phenomena for our hypotheses in Appendix A. ¿From this analysis we can see that hypothesis H_1 will always be preferred over hypotheses H_0, H_2 and H_3, and the same holds for the hypotheses belonging to the other dimensions (i.e., H_5 will be preferred over H_0, H_6 and H_7, etc.). Therefore, without penalties for dividing an area into multiple areas, the program would usually tend to divide each volume to two new volumes, until each voxel is represented separately.

To overcome this problem, we use the method developed in [4] to incorporate into the search algorithm our prior expectations regarding the number of anomalies, their sizes and shapes. This is done by introducing a set of *penalties* which are used just like a set of prior-probabilities for the different hypotheses. More precisely, for the hypothesis testing we use a MAP procedure, which requires that we have prior probabilities for the hypotheses. Recall from Eq. 2.5 that the MAP decision rule is

$$m = \text{argmin}_j \frac{1}{2}(\mathbf{y} - \mathbf{B}_j\mathbf{a}_j)^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{B}_j\mathbf{a}_j) + \pi_j$$

We use the penalty values in the role of these π_j's. These penalty terms are computed in our algorithm as

$$\pi_j = 10^{\gamma} \cdot f_j \tag{3.1}$$

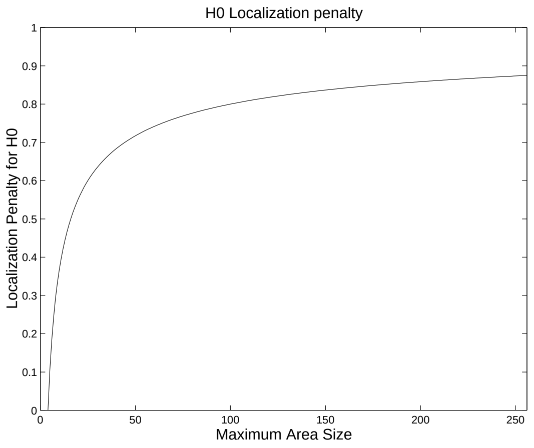
The term γ in the above expression is a constant which is computed once at the beginning of the algorithm, and remains fixed throughout. This constant determines the relative magnitude of the penalty terms in the MAP decision rule from above. Thus, if γ is too small then the penalty terms will be negligible and will not affect the tendency of the algorithm to over-fit the data, and if γ it is too large then the contribution of the data itself will be negligible. Following [4], we set this constant to be

$$\gamma = \left\lfloor \log_{10} \frac{1}{2}(\mathbf{y} - \hat{a}\mathbf{1})^T \mathbf{R}^{-1}(\mathbf{y} - \hat{a}\mathbf{1}) \right\rfloor - 1$$

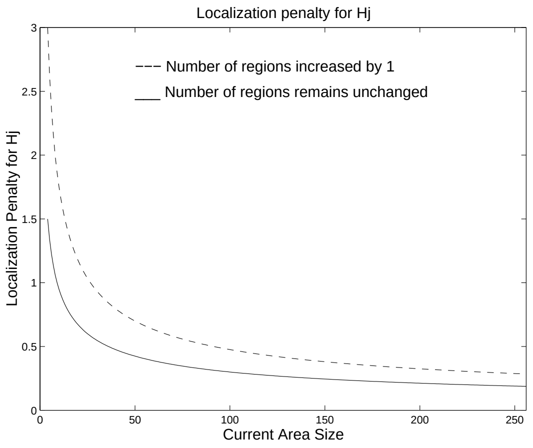
where $\mathbf{y}$ is the observed vector, $\mathbf{1}$ is the all-one vector, and $\hat{a}$ is the ML estimate of the amplitude for an anomaly which covers the whole volume. This setting ensures that all the π_j's are roughly one order of magnitude smaller than the normalized variance of the data. (We stress that this is merely a heuristic setting which seems to work well in this algorithm.)

The term f_j in Eq. 3.1 is the penalty function for hypothesis H_j, and is described next. We have four types of penalty functions: two affect localization and two affect pruning. The general shape of these functions follows our expectations for the characteristics of the anomaly structure, and is explained below for each function. In particular, all these functions are monotone in their arguments (so, for example, the penalty is either monotonically increasing with the size of the region or monotonically decreasing with this size). The exact shape of these functions were tuned via experiments to obtain the best performance.

These penalty functions were tuned for the volume size which we use in our experiments, namely 16^3. Since the algorithm is scale-sequential, then we know that these functions also work well for smaller volumes. This is since as the algorithm advances, these smaller volumes are actually encountered. For larger volumes, it seems reasonable to assume that these penalty functions should provide similar results for similar SNR levels, since neither the penalty functions themselves nor the hypotheses likelihoods depend on the overall volume size. This assumption, however, may be subject to future testing. It should be noted that the penalty functions could be changed according



(b)



(c)

Figure 3.5: Localization penalty functions for $MAS = 4$

to different scenarios in different applications, to meet different prior expectations. The main contribution of these functions is to demonstrate the ability to introduce prior information into the algorithm to guide the search.

3.4.1 Localization Penalties

The localization penalties reflect our expectations that the anomalies are not too small neither too large, and that we do not expect too many anomalies in the region. We have two penalty functions in the localization step, one for computing f_0 (i.e., the penalty term for keeping the whole region without further sub-division) and the other for computing the other f_j's (i.e., the penalties for further zooming in into a region and refining of the current structure).

H_0 **penalty** (f_0) Penalty function for choosing H_0 (Halting): The function is monotonic, and grows with the size of the largest current region. This reflects the fact that we do not expect to have very large anomalies, therefore we examine the current largest anomaly, and the larger it is, the larger the penalty is. This ensures that we will not have a region which is too big. To compute this penalty, we use the following quantities.

1. MAS (Min Area Size). This is a parameter which represents the smallest acceptable size for an anomaly. This parameter is set before we start the search. In our tests we defined $MAS = 4$ (namely, we did not allow the program to output any anomaly which is smaller than four voxels).

2. LAS (Largest Area Size). The size of the largest area currently on the list.

We compute the H_0 localization penalty as

$$f_0 \overset{\text{def}}{=} 1 - \sqrt{MAS/LAS} \tag{3.2}$$

See Fig. 3.5 (a) for a plot of f_0 vs. LAS when $MAS = 4$.

H_j **penalty** (f_j, $j = 1, \ldots, m$). Penalty function for not halting: When we choose any hypothesis other than H_0, this hypothesis dictates that we divide one of the current regions into one or more new regions. This penalty function depends on the size of the new regions and how many new regions are there.

The function f_j grows as the size of the new regions in the configuration decreases. This is done since we want to override the tendency of the program to over-fit the data, and so we want to prevent it from finding many small anomalies. In our implementation we set a minimum size of an anomaly, the closer we are to this size, the larger the f_j penalty would be. The function also grows with the number of new regions in the configuration. Since we do not want to have too many areas, we have a larger penalty value for divisions which increase the number of regions on the current list.

There are some cases in which f_j assumes the value $+\infty$. In particular, since we expect that the anomalies are "not too small and not too many", we do not allow the algorithm to keep regions which are smaller than the Min-Area-Size parameter (i.e., four voxels) or to keep more than ten regions on its list at the same time. Therefore, the penalty for choosing a hypothesis which violates these constrains is defined to be infinite. Also, we do not allow the algorithm to keep any region in which the length in one dimension is greater then four times the length in another dimension. Therefore, we have an infinite penalty for any division which results in such a region. For example, if a region in the current list is of size $16 \times 8 \times 4$, then we associate an infinite penalty with the subdivision which result in a region of size $16 \times 8 \times 2$. We also limit the size of the minimum anomaly to be 4 voxels. To compute the f_j functions, we define the following quantities:

- The minimum-area-size parameter (MAS) is defined as above.
- The size of the regions which result from choosing H_j is denoted by CAS_j. (Note that in our hypotheses, all the regions have the same size.)
- The ratio between the largest and the smallest side-length in the regions which result from choosing H_j is denoted by MaxRatio_j.
- The number of regions in the current list after choosing hypothesis H_j is denoted by M_j.
- We define Δ to be the difference between the number of regions on the list before and after choosing H_j. Namely

$$\Delta_j \stackrel{\text{def}}{=} M_j - M_0$$

(Notice that since H_0 i is the null hypothesis then M_0 is indeed the number of regions before choosing any hypothesis).

- The penalty 'Per-Added-Region' for hypothesis H_j is defined as

$$PAR_j \overset{\text{def}}{=} 0.25 + 0.75 \cdot \sqrt[4]{MAS/CAS_j}$$

and then we compute the f_j localization penalty as

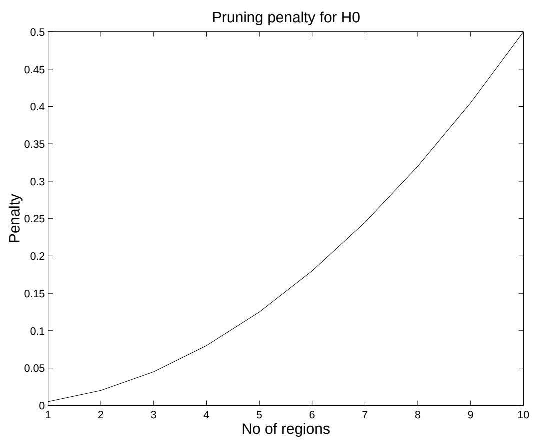
$$f_j \overset{\text{def}}{=} \begin{cases} +\infty & \begin{array}{l}\text{If } CAS_j < MAS \\ \quad \text{or MaxRatio}_j > 4 \\ \quad \text{or } M_j > 10\end{array} \\ \frac{3}{2} \cdot \sqrt{MAS/CAS_j} \cdot (1 + \Delta_j \cdot PAR_j) & \text{otherwise} \end{cases} \tag{3.3}$$

As can be seen, if the number of regions on the list does not change when choosing H_j, then $\Delta_j = 0$, and thus PAR_j is never used. See Fig. 3.5 (b) and (c) for plots of this penalty function vs. LAS for $\Delta \in \{0, 1\}$ and $MAS = 4$.

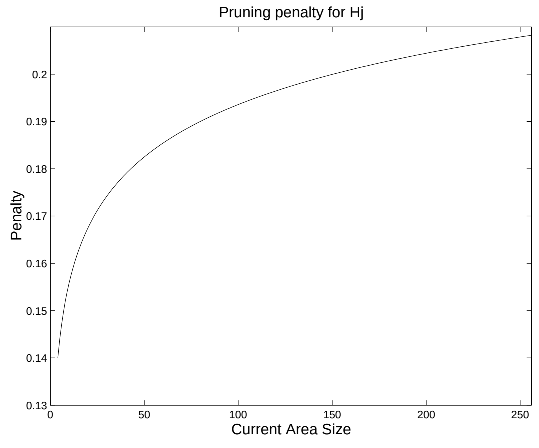
3.4.2 Pruning Penalties

The pruning penalties incorporate our expectations that there are not too many anomalies in the volume, and the fact that the algorithm is more likely to "miss an anomaly" in large regions than in small ones. Therefore we prefer to keep large regions for further refinement even if it currently seems that they do not contain anomalies, rather than to prune them right away. Here too we have two types of penalty functions - one for not pruning anything and one for pruning something.

H_0 **penalty** (f_0) Penalty function for not pruning away any anomaly: This is a function of the current region configuration, and grows with the



(a)



(b)

Figure 3.6: The pruning penalty functions: (a) H_0 penalty; (b) H_j penalty.

number of regions in the configuration. We do not want to have too many areas. Therefore, the more areas we have, the bigger the penalty is for not pruning any of them. To compute this function we denote the number of anomalies in the current configuration by NR and compute

$$f_0 \overset{\text{def}}{=} \frac{1}{2}\left(\frac{NR}{10}\right)^2 \tag{3.4}$$

See Fig. 3.6 (a) for a plot of this penalty function vs. the number of areas in the current list.

H_j **penalty** ($f_j,\ j = 1, \ldots, m$) Penalty function for pruning an area: When we choose any hypothesis other than H_0, this hypothesis dictates that we prune one of the current regions. The penalty for this choice is a function only of the pruned region. It grows as the size of the pruned area increases, because the algorithm has a larger probability of missing a small anomaly in a large region.

To compute this function we use the MAS parameter (Min-Area-Size) and denote by CAS_j the size of the region pruned by the hypothesis H_j.

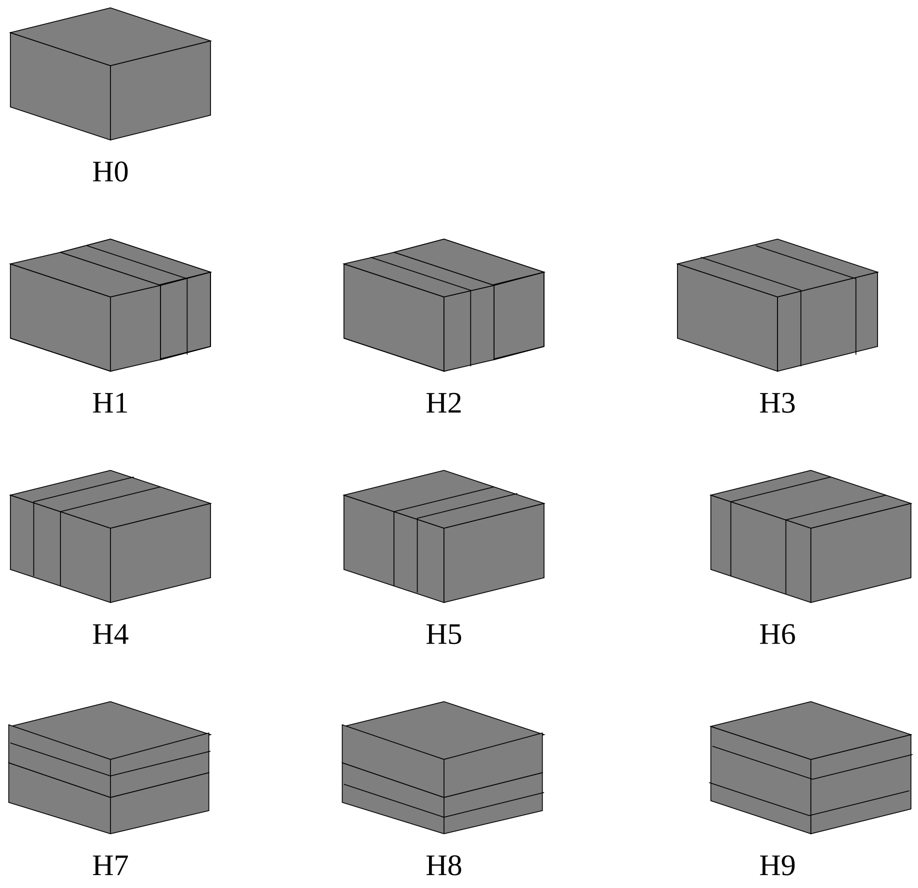


Figure 3.7: Initial hypotheses

Then we set

$$f_j \overset{\text{def}}{=} 0.7 \cdot \left(1 - 0.8 \cdot \sqrt[32]{MAS/CAS_j}\right) \tag{3.5}$$

(We stress that the expression $\sqrt[32]{\cdots}$ was chosen via experiments. The value of 32 simply gave the best results in our tests.) See Fig. 3.6 (b) for a plot of this penalty function vs. CAS.

3.5 Initial Hypotheses

In our algorithm we use a different set of hypotheses for the first division of the area. The reason is that during the initial processing stage, the chances of missing an anomaly are relatively high since the areas involved are rather large. In [5] it was shown that the primary difficulty associated with the general algorithm is that coarse scale detection probabilities can be low. Therefore, at this stage we keep more coarse-scale regions for further examinations. The set of hypotheses which we use at this stage is described in Fig. 3.7. Notice that all these hypotheses divide the area into a few regions, and we do not drop any part of the area. The initial hypotheses are described graphically in Fig. 3.7.

3.6 Feedback

One problem with localizing the anomalies accurately is that we have only a few specific ways to divide the region in each step. Thus, the chances of "cutting away" parts of the anomalies are rather high, especially at initial stages when the size of the examined regions is large relative to the size of the anomalies. Another problem is when the amplitude of different anomalies are of different magnitude. In this case, an anomaly that is much larger may "mask" the smaller anomalies.

In order to improve the accuracy of the algorithm, we therefore implemented a collection of "feedback methods". Namely, we view the results of the first run of the algorithm as a rough estimate for the anomaly configuration, and then use these results to guide us in subsequent runs. We use two feedback methods in our algorithm.

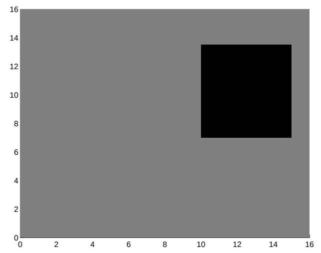
3.6.1 Reexamining Found Anomalies

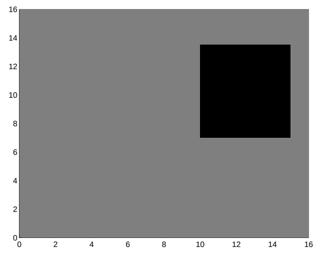
After locating the anomalies, we reexamine the close neighborhoods of these anomalies to obtain a more accurate localization. This process is illustrated in Fig. 3.8(e)-(g). We first add a "frame" around each anomaly, which is centered in the center of this anomaly and has width and height twice as the anomaly itself. Inside this frame, we then run again our search algorithm "from scratch". We run this process separately for each anomaly. We will refer to this feedback method as *Reexamining feedback*.

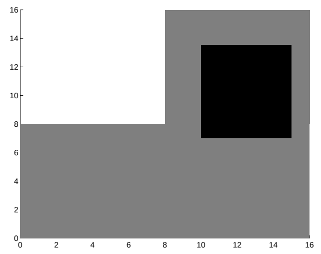
Notice that this feedback process may result in larger anomalies than before the feedback phase. Hence, it raises the possibility of two anomaly areas overlapping. For simplicity, we discard any anomaly which overlaps another previously found anomaly. For example, if after running this feedback procedure on the first two regions we find that the resulting two regions overlap, then we discard the second region from our anomaly structure.

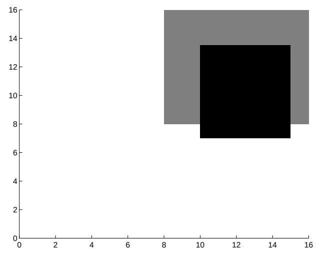
The reason that we expect this feedback method to improve the accuracy of the localization is as follows. At the first stages of the algorithm, the regions examined are typically large relatively to the sizes of the anomalies. Thus, the algorithm may "cut-off" a part of the anomaly when choosing a hypothesis which keeps most of the anomaly. In our experiments, we found that this method significantly improves the accuracy of the anomaly borders and amplitudes.

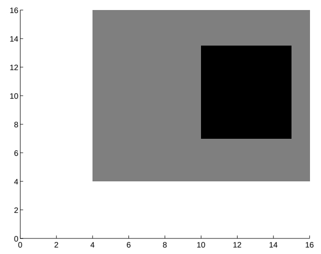
This feedback method is demonstrated in Fig. 3.8. In (a)-(d) we see the localization and pruning, and in (e)-(g) we see that a frame is added, and the localization algorithm is being run again, resulting in a more accurate solution.

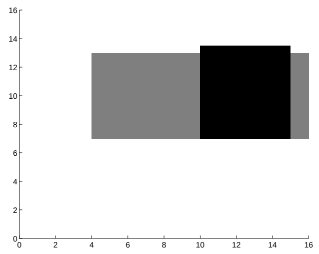

(a) Initial configuration


(b) First localization


(c) Second localization


(d) Result before feedback


(e) Adding the frame


(f) Localization step

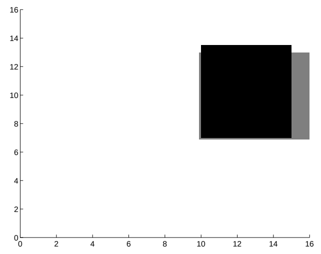

(g) Final result

Figure 3.8: Example of execution with feedback. The black region is the anomaly and the gray regions are the hypotheses of the algorithm.

3.6.2 Subtracting Data of Found Anomalies

Experiments with our algorithm exhibits problems when the region contains anomalies of different orders of magnitude. In this case, the small anomalies are negligible relative to the large ones and are often ignored by the algorithm. Our experiments show that the algorithm finds the smaller anomalies when we subtract from the data the effects of the found anomalies and rerun the program on the new data. Therefore, when we expect to have anomalies with different orders of magnitude, we incorporate into the program this feedback. We will refer to this type of feedback as *Subtracting feedback*.

This method is also used when the number of found anomalies is large. The reason is that due to our penalties, the program can find only a limited number of anomalies in each iteration. If running the program in one iteration finds relatively many anomalies, then there might be more, which will be found if we were to subtract the found anomalies from the data and run the program again.

3.7 Final Processing

At the end of the program, we go over all the found regions and eliminate those which are likely to be false alarms. To this end, we use the LRT decision rule for binary hypothesis testing. For every region we have the two hypotheses

$$\begin{aligned} H_0 &: \mathbf{y}' = \mathbf{n} \\ H_1 &: \mathbf{y}' = \hat{a} \cdot \mathbf{1} + \mathbf{n} \end{aligned}$$

where $\mathbf{y}'$ is the partial vector consisting only of the data in the region whose indicator vector is B_i, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, $\mathbf{1}$ is the all-1 vector over this region and $\hat{a}$ is the estimated amplitude of this region, namely $\hat{a} \overset{\text{def}}{=} \frac{1}{N} \sum_i y'_i$. Repeating the derivation from Section 2.1.3, the LRT rule for this case can be written as

$$\hat{h}_{LRT}(\mathbf{y}) = \begin{cases} 0 & \text{if } \frac{(\mathbf{y}')^T\mathbf{y}'-(\mathbf{y}'-\hat{a}\mathbf{1})^T(\mathbf{y}'-\hat{a}\mathbf{1})}{2\sigma^2} \leq \tau \\ 1 & \text{otherwise} \end{cases}$$

We can re-write the expression on the right hand side of the LRT rule as

$$\begin{aligned} &\tfrac{1}{2\sigma^2}[(\mathbf{y}')^T\mathbf{y}' - (\mathbf{y}' - \hat{a}\mathbf{1})^T(\mathbf{y}' - \hat{a}\mathbf{1})] \\ &= \tfrac{1}{2\sigma^2}\textstyle\sum_{i=1}^{n}[y_i^2 - (y_i - \hat{a})]^2 \\ &= \tfrac{1}{2\sigma^2}\left(2\hat{a}(\textstyle\sum_{i=1}^{n} y_i) - N\hat{a}^2\right) \\ &= \tfrac{1}{2\sigma^2}\left(2\hat{a}\cdot n\hat{a} - n\hat{a}^2\right) \qquad = (n\hat{a}^2)/(2\sigma^2) \end{aligned}$$

(where n is the size of the region at hand). Thus, the LRT rule consists of comparing the sufficient statistic $\frac{Na^2}{2\sigma^2}$ to the threshold τ. In our experiments we worked with $\sigma = 1$ and set the threshold to $\tau = 8$. In the final processing, therefore we discarded any region for which $n\hat{a}^2 < 16$ (where n is the size of the region and $\hat{a}$ is the average amplitude in the region).

For this decision rule it is also easy to compute the probability of false-alarm. Since according to H_0, we have $\mathbf{y}' \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, and since $\hat{a} = \frac{1}{N}\sum_{i=1}^{N} y_i$, then $\hat{a}$ is a Gaussian random variable $\hat{a} \sim \mathcal{N}(0, \frac{\sigma^2}{N})$. Thus we get

$$P_f = \Pr\left[\frac{n\hat{a}^2}{2\sigma^2} > \tau \mid H_0\right] = \Pr\left[|\hat{a}| > \sigma\sqrt{\frac{2\tau}{n}} \mid H_0\right] = 2\cdot Q\left(\sqrt{2\tau}\right)$$

Where the Q function is defined

$$Q(x) \overset{\text{def}}{=} \frac{1}{\sqrt{2\pi}}\int_x^\infty \exp\left(\frac{-t^2}{2}\right) dt$$

Thus, the value of $\tau = 8$ corresponds to false-alarm probability $P_f = 2\cdot Q(4) \approx 5\cdot 10^{-4}$.

3.8 Generalization

In this work we consider a model in which the data consists of the intensity field and additive white Gaussian noise. However, this algorithm can be used for the model in which the measured data is a function of the intensity field with additive white Gaussian noise. As described in [4], the model in this case will be

$$\mathbf{y} = \mathbf{f}(\mathbf{g}) + \mathbf{n}$$

In this case, each hypothesis H_j will be described as following:

$$H_j : \mathbf{y} = \mathbf{f}(\mathbf{B}_j \hat{\mathbf{a}}_j) + \mathbf{n} \tag{3.6}$$

and the maximum-likelihood estimator of the amplitude will be:

$$\hat{\mathbf{a}}(\mathbf{y}) = \mathrm{argmin}_{\mathbf{a}}[(\mathbf{y} - \mathbf{f}(\mathbf{Ba}))^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{f}(\mathbf{Ba}))]$$

The MAP procedure employed to choose from these hypotheses the hypothesis H_m will be of the form:

$$m = \mathrm{argmin}_j (\mathbf{y} - \mathbf{f}(\mathbf{B}_j \hat{\mathbf{a}}_j))^T \mathbf{R}^{-1}(\mathbf{y} - \mathbf{f}(\mathbf{B}_j \hat{\mathbf{a}}_j)) + \pi_j \tag{3.7}$$

Thus, we can still use our algorithm to built the $\mathbf{B}_j$ matrices corresponding to our hypothesis, and choose from them using the MAP procedure.

3.8.1 The Linear Case

In the case in which the function of the intensity field is a *linear function* (namely, $\mathbf{f}(\mathbf{g}) = \mathbf{Tg}$ for some matrix $\mathbf{T}$), the maximum-likelihood estimator of the amplitude is given by

$$\hat{\mathbf{a}}(\mathbf{y}) = \left(\mathbf{B}^T \mathbf{T}^T \mathbf{R}^{-1} \mathbf{T} \mathbf{B}\right)^{-1} \cdot BB^T \mathbf{T}^T \mathbf{R}^{-1} \mathbf{y} \tag{3.8}$$

In this case, the matrix $\left(\mathbf{B}^T \mathbf{T}^T \mathbf{R}^{-1} \mathbf{T} \mathbf{B}\right)$ is typically not a diagonal matrix, so we will need to perform matrix inversion for each hypothesis test.

Chapter 4

Complexity Analysis

4.1 Hypotheses Testing Complexity

In this chapter we give a partial analysis for the hypothesis test complexity of our algorithm. We start by presenting a simple upper bound on the worst-case behavior of the main search routine of our algorithm (i.e., no feedback) *without the pruning phases.* We next analyze the worst case behavior of this procedure including the pruning. Then we discuss briefly the complexity of the feedback mechanisms, and compare our complexity bounds with the complexity of an exhaustive search.

It is interesting to note that this analysis yields a much better upper bound for the case with no pruning. Moreover, our tests show that the actual behavior of the algorithm is much closer to the bound for the case with no pruning. Indeed, in our experiments, only a relatively small amount of pruning occurred. We note that the number of regions is controlled not only by the pruning stages, but also (in fact, mostly) by the localization penalties, which do not allow the number of regions to grow beyond a specific bound. All these considerations raise the possibility that the pruning steps may in fact be redundant. However, we did not test the algorithm without the pruning. This may be subject for future work.

4.2 Complexity with no Pruning

Recall that the main search routine of the algorithm works by maintaining a list of current regions, and at each step either splitting one of the regions in this list (in a localization step) or dropping it altogether (in a pruning step). In the analysis below we assume that no pruning occurs, so in every step the algorithm simply takes one region from the list and split it in two, keeping either one or both of the halves on the list. Below we refer to the list of regions as the *current list*. During a particular execution of the algorithm, we consider the following quantities

- The number of voxels in the whole medium is denoted by N.
- The number of localization hypotheses for each region, which in our implementation is 12.
- The maximum number of regions that were simultaneously kept on the current list at any point during the execution is denoted by M. Our implementation prevents the algorithm from keeping more than 10 regions at the same time, so in every execution we have $M \leq 10$.

We now proceed to bound the number of hypothesis tests in this execution in terms of these quantities. Recall from Section 3.1 that we can view the execution of the algorithm via the notion of an "execution tree" which consists of all the regions that were on the current list at any time during the execution. At the root of this tree we have a region which consists of the whole medium, and we say that region A is a parent of region B in the tree if B was placed on the current list by splitting A during one of the localization steps. We can view each step in this execution as picking one of the leaves in the tree and splitting it, keeping either one or two of its children. When the algorithm finally halts, we can make the following observations about the execution tree:

Lemma 4.1 *If the algorithm does not perform pruning, then the number of nodes in the execution tree is at most $M \log_2 N$.*

Proof: Define a *level* in the tree to be all the nodes at a certain height above the leaves. Namely, a node is at level i if the distance from this node to the nearest leaf is i (so all the leaves are at level 0, their parents are at level 1, etc.). Then we can do the following observations:

1. Since the size of each region in the execution tree is exactly half that of its parent, and since the size of the root is N, then the depth of the tree is at most $\log_2 N$. Hence, the tree contains at most $\log_2 N$ levels.

2. Denote the number of nodes in level i in the tree by L_i. Since every node in level $i > 0$ in the tree has at least one child in level $i-1$, then for all $i > 0$ we have $L_i \leq L_{i-1}$.

3. Since we never keep more than M regions on the current list at the same time, then in particular at the end of the execution we keep at most M leaves. Moreover, if the algorithm does not perform pruning then this means that the entire execution tree never contained more than M leaves (as no leaf is never pruned, and thus every leaf in the execution tree must be on the current list when the algorithm halts). Therefore we have $L_0 \leq M$.

Combining the last two observations we conclude that every level in the tree consists of at most M nodes. As there are only at most $\log_2 N$ levels, the tree cannot contain more than $M \log_2 N$ nodes. ■

Theorem 4.2 *If the algorithm does not perform pruning, then the number of hypothesis-tests in every execution is at most* $(12M + 1) \cdot M \log_2 N = O(M^2 \log N)$.

Proof: Note that every region in the execution tree enters (and leaves) the current list exactly once during the execution. Hence, the number of steps in the execution is bounded by the number of nodes in the execution tree. Since in each step during the execution we consider at most $12M + 1$ hypotheses in the localization stage, then the total number of hypotheses in this execution is bounded by $(12M + 1) \cdot M \log_2 N$. ■

Remark: The Size of M. In our implementation we limit M to be ten, by not letting the program to take two halves of a region if we have already ten regions. However, in practice, for a small number of anomalies, M does not reach this bound, and for the case of one or two anomalies, M is usually much smaller then ten regions.

4.3 Complexity with Pruning

If we allow the algorithm to prune leaves, then the above upper bound no longer holds in the worst case. In particular, consider the following scenario which is depicted in Fig. 4.1: In the first step, the algorithm splits the original region and keeps both halves. Then, it keeps splitting the right half until it has $M - 1$ regions on the right, while leaving the left half untouched. Next the algorithm continues to split the $M - 1$ regions on the right, each time leaving only one of the halves, so that the number of regions on the current

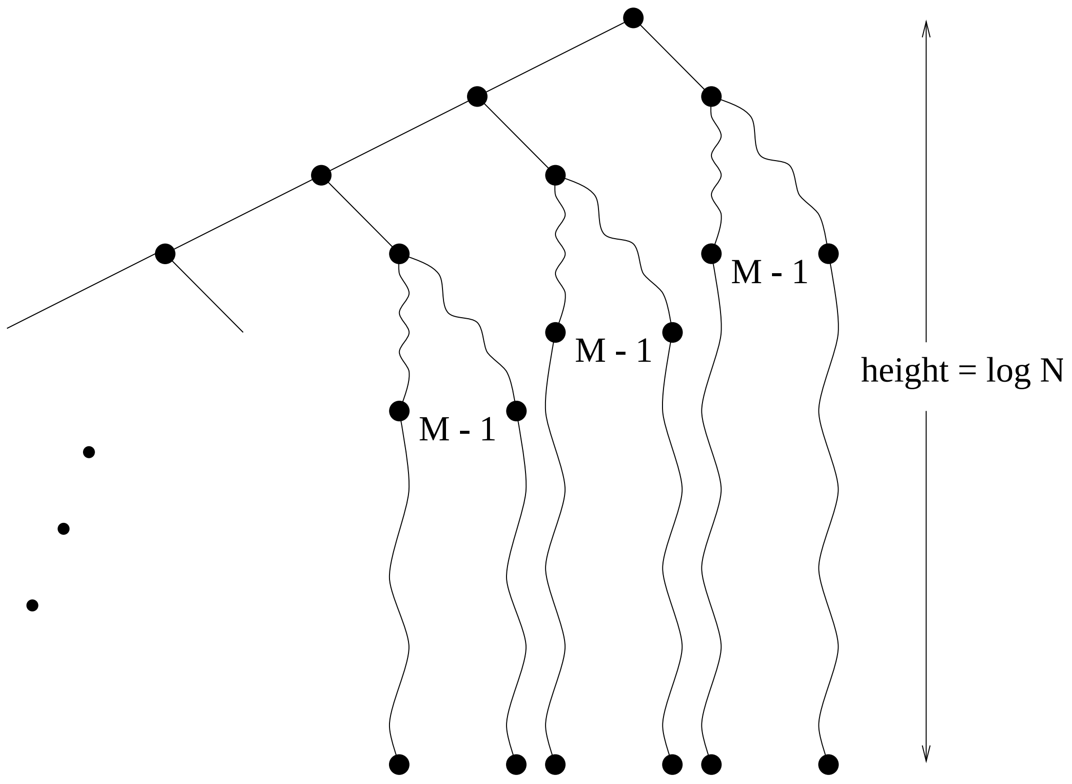


Figure 4.1: An example of an execution tree with many nodes

list never grows beyond M. This continues until all these $M-1$ regions reach the minimum size, at which point the algorithm prunes all of them. Now the algorithm is left with a single region of size $\frac{N}{2}$, and it can repeat the above process starting from this one region. The number of regions in the resulting execution tree is about

$$\#\text{-of-nodes} \approx (M-1)\log N + (M-1)\log(\tfrac{N}{2}) + (M-1)\log(\tfrac{N}{4}) + \ldots$$
$$= \textstyle\sum_{i=1}^{\log N} i(M-1) \approx \frac{(M-1)\log^2 N}{2}$$

In fact, even the above example is not a worst-case behavior of the algorithm. In Appendix B we prove that the worst-case complexity with pruning is $O(M\binom{M+\log_2 N}{M})$. In reality, however, the algorithm does very little pruning, and in Chapter 5 we show that the complexity for a typical anomaly configuration is indeed logarithmic in N. Therefore, for the purpose of comparison with exhaustive search, we use the bound which we obtained for the case with no pruning.

4.4 Comparison with Exhaustive Search

To analyze the performance of an exhausting search, we notice that a rectangular volume can be specified by two of its corners. Therefore, in order to find a single rectangular anomaly of unknown size within a medium of volume N by an exhaustive search, we need to consider about $\binom{N}{2} \approx \frac{N^2}{2}$ possibilities.

If we have some prior knowledge about the volume of this anomaly, then this work can be reduced. For example, if we know a-priori the exact shape of the anomaly then we only need to find one of its corners, so we have only N possibilities to consider. If we do not know the exact shape, but we know that it can be one of k different shapes, then the complexity is $O(kN)$. For example, if we know that the volume of the anomaly is at most v, then we only need to consider those shapes with volume up to v. If we denote by $k(v)$ the number of different shapes (length $\times$ height $\times$ depth) with volume up to v, then we have complexity of about $N \cdot k(v)$. We can approximate the value of $k(v)$ using the integral

$$k(v) \approx \int_{x=1}^{v} dx \int_{1}^{v/x} dy \int_{z=1}^{v/xy} dz \ = \frac{v \ln^2 v}{2} - v \ln v + v - 1 \ \approx \ \frac{v \ln^2 v}{2}$$

(this estimate is only valid for $1 \ll v \ll N$).

In any case, it is obvious that even finding a single anomaly of known size via exhaustive search requires work which is at least linear in N, as opposed to logarithmic in N using our algorithm.

Suppose now that we want to find M anomalies by exhaustive search. In this case, we need to localize all the anomalies at once, and since there are about $N \cdot k(v_i)$ possibilities to localize the i'th anomaly (where v_i is the size of that anomaly), then the total hypotheses tests count will be about $\prod_{i=1}^{M}(N \cdot k(v_i))$.

4.4.1 A Numerical Example

Assume that we are looking for an anomaly whose volume is at most 6 voxels within a volume of $N = 16 \times 16 \times 16 = 4096$ voxels. A simple counting shows that $k(6) = 1 + 3 + 3 + 6 + 3 + 6 = 22$ (since we have one shape with volume 1, three shapes with volume 2, etc.). Thus we need to consider about $22 \times 4096 \approx 90000$ possibilities.

In our algorithm, on the other hand, if we assume that $M = 7$ (which is a reasonable assumption for the case of a single anomaly, according to our tests), then the number of hypotheses which are considered during an execution is bounded by $(12 \times 7 + 1) \times 7 \times \log_2 4096 \approx 7100$.

We note that the above estimation is in fact quite generous. For one thing, we assume that a good approximation for the volume of the anomaly is known ahead of time. Moreover we assume that there is only a single anomaly to find. In fact, our algorithm can find several anomalies in each run, whereas the above exhaustive search routine can only find one anomaly at a time. Finally, the hypotheses tests estimation for the performance of our algorithm is overly pessimistic, and the actual number of hypotheses is likely to be smaller than this bound (see the test results in Chapter 5).

4.4.2 Feedback Complexity

For the first type of feedback, which consists of reexamining the close neighborhood of the found anomalies, the complexity depends on the size of the found anomalies. Since usually the found anomalies are small relative to the size of the whole region, this type of feedback still stays in the magnitude of $O(\log N)$. Specifically, if before the feedback we have one anomaly with volume v, then the complexity of the feedback stage will be $O(\log v)$, and the total complexity will be $O(\log N) + O(\log V) = O(\log N)$. In our tests, the reexamination feedback typically added 20 percent to the number of hypotheses tests and in the worst case the complexity was increased by 50 percent.

The second type of feedback consists of re-running the program again on a new data. When this is executed, then the number of operations needed to execute the whole program roughly multiplies. In our implementation, this feedback is only done if we found a large number of anomalies in the first run. (Note that in this case, the exhaustive search complexity will also be much larger.)

Chapter 5

Test Results

The program was tested on several instances of synthetically generated data. The overall volume on which the program was tested was chosen to be of size $16 \times 16 \times 16$ voxels. This volume is large enough to admit many different types of anomaly-structures, yet small enough so that the program still runs very fast. This enabled us to repeat the tests many times for each data point, so we can obtain meaningful estimation of the probabilities of detection and false-alarm.

Each data point in our tests consists of a specific anomaly-structure and a specific SNR level. The SNR level is computed as follows: Let $\mathbf{g} = \mathbf{Ba} = \sum_i a_i \mathbf{b}_i$ be the anomaly structure, and let $\mathbf{y} = \mathbf{g} + \mathbf{n}$, be the observed data, where $\mathbf{n}$ is a white Gaussian noise $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 I)$. The signal-to-noise ratio (SNR) of $\mathbf{y}$ (in decibels) is

$$\mathrm{SNR} \stackrel{\mathrm{def}}{=} 10 \log_{10} \frac{\mathbf{g}^T \mathbf{g}}{N\sigma^2} \text{ dB}$$

where N is the number of entries in $\mathbf{y}$. Below we refer to this number as the *total-SNR* of this data point. Also, the SNR of the i'th anomaly (which is described by the vector $a_i\mathbf{b}_i$) is

$$\mathrm{SNR}_i \overset{\mathrm{def}}{=} 10\log_{10}\frac{a_i^2(\mathbf{b}_i^T\mathbf{b}_i)}{N\sigma^2}$$

Below we refer to this number as the *anomaly SNR*. We can compute the anomaly SNR as $\mathrm{SNR}_i = 10\log_{10}\frac{(a_i^2 N_i)}{(\sigma^2 N)}$, and since in our tests we used $\sigma = 1$, then we have $\mathrm{SNR}_i = 10\log_{10}\frac{a_i^2 N_i}{N}$, where N_i is the volume of the i'th anomaly. We vary the SNR by varying the anomaly amplitudes.

In our experiments, we define a *detection* to be any region identified by the algorithm for which at least 25% of its volume overlaps a true anomaly. Anything else is considered a *false alarm*. For each anomaly we compute the "empirical P_d" as

$$\bar{P}_d \overset{\mathrm{def}}{=} \frac{\text{Number of times anomaly was detected}}{\text{Total number of runs}}$$

We then computed for every test the "empirical P_f" as

$$\bar{P}_f \overset{\mathrm{def}}{=} \frac{\text{Total volume of falsely detected regions}}{\text{Total volume of the entire medium}}$$

We run one hundred tests for each data point (each time adding independently chosen noise), and computed the P_f and P_d values of this data point as the average of the empirical P_f and P_d values, respectively, of all these tests. In all the tests that we performed, we set the threshold value τ of the post-processing stage so as to obtain $P_f = 5 \cdot 10^{-4}$ (See Section 3.7).

In the description below, we specify the location of an anomaly by a vector $(\langle x_1, y_1, z_1\rangle, \langle x_2, y_2, z_2\rangle)$, of which $\langle x_1, y_1, z_1\rangle$ is the front-lower-left corner of the anomaly and $\langle x_2, y_2, z_2\rangle$ is the back-upper-right corner. For example, a region which is specified by $(\langle 2,4,6\rangle, \langle 5,9,7\rangle)$, is located in the x direction between voxel 2 and 5 (including voxels 2 and 5), in the y direction between voxels 4 and 9, and in the z direction between voxels 6 and 7.

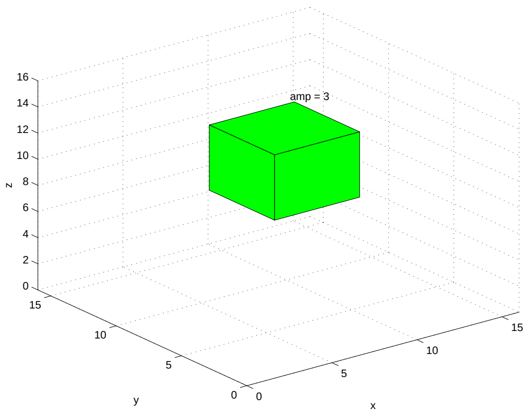
For each test we measured the number of hypothesis tests before and after the reexamining feedback. The P_d and P_f were measured at the end of the program, after the reexamining feedback. However, the P_d and P_f were essentially the same before the reexamining feedback. This feedback only helps to localize more accurately the anomaly. If we changed the detection definition so that a detection will be a region which at least 75% of its volume overlaps a true anomaly (instead of 25%), then we would expect a change in the P_f after the reexamination feedback, since in this case before the feedback some of the regions will be considered false alarms.

5.1 A Single Anomaly

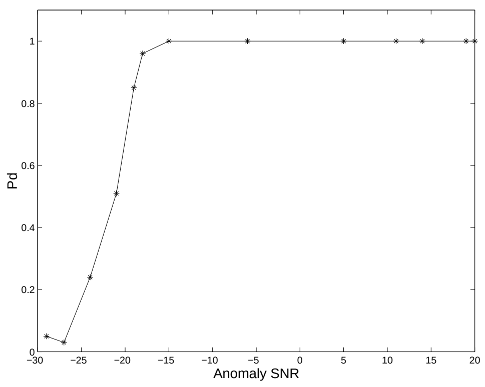
We first considered an anomaly configuration which consists of a single anomaly. Within the total volume of 16^3 voxels, we placed a single anomaly of size 5^3 near the middle of the region, in location $(\langle 7,7,7\rangle, \langle 11,11,11\rangle)$. The location and size of the anomaly are demonstrated in Fig. 5.2(a). For this anomaly configuration we tested SNRs between -30 and 20dB (since we have a single anomaly, then the total SNR and the anomaly SNR are the same).

The calculated false alarm probabilities were between 0 and $2.5 \cdot 10^{-4}$ throughout the entire range. The resulting P_d values for different SNRs are summarized in Fig. 5.1(b). As we can see, the detection probability is very close to 1 for SNRs as low as -17dB.

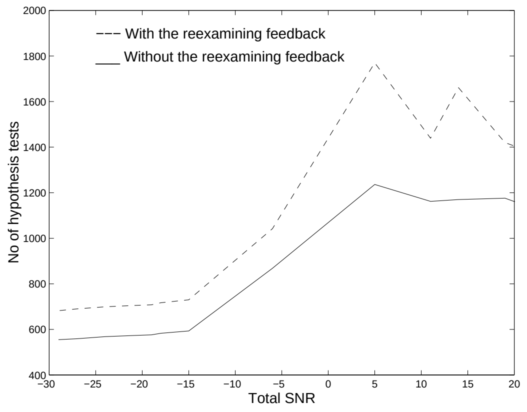
In addition to the P_d and P_f levels, we also measured the hypothesis testing complexity of these experiments. The complexity is plotted in Fig. 5.1(c) against the SNR level. To give a feeling for the cost of the feedback, we measured the hypothesis testing complexity with and without the reexamining feedback. As can be seen in the figure, this type of feedback increases the



(a) Input configuration



(b) P_d vs. SNR



(c) Complexity vs. SNR

Figure 5.1: Test results for a single anomaly

hypothesis testing complexity by 20-50%. Also, the figure demonstrates that in practice, the algorithm performs at most 16% of the number of hypothesis tests which are predicted by the analytic upper-bound in Chapter 4. While the numerical example in our analysis yields a bound of about 7100 hypothesis tests (without the feedback), the actual performance was always less than 1250 hypothesis tests.

We note that although the bound from Chapter 4 is independent of the SNR level, our experiments indicate that the SNR level influences the M parameter in this bound (i.e., the maximum number of regions in the current-list at any given time). Indeed, it can be seen from Fig. 5.1(c) that the hypothesis testing complexity increases significantly for higher SNR levels. This behavior can be explained as follows: Recall that the algorithm uses only a small number of hypotheses in every step. This behavior limits the number of shapes that the regions on the current-list can take. (For example, in our case the side length of every region on the current-list must be a power of two.) In the tests above, however, the actual anomaly was of size $5 \times 5 \times 5$, and the algorithm typically quickly finds a piece of it of size $4 \times 4 \times 4$. For low levels of SNR, the remaining pieces of the anomaly are ignored, since they are too small to be noticed. As the SNR increases, however, these small pieces become more and more noticeable, and so the algorithm ends up keeping all of them on the current list, hence increasing the number of hypotheses which need to be considered at each step.

Finally, in Figures 5.2 and 5.3 we show a few snapshots from the execution of the algorithm on this anomaly configuration with SNR of -6dB. The shades in these figures are relative to the intensity of the anomalies: Bright regions denote large amplitudes, while dark regions denote small amplitudes. Fig. 5.2(a) describes the original anomaly structure. In (b) we see the first partition, using the initial hypothesis. As we can see in (c), the algorithm first zooms-in on the right anomaly. However, it keeps the other two regions for further checking, since smaller anomalies might still exist in them. In (d) we see the partitioning of the rest of the areas, and (e) shows the end of the algorithm before the feedback.

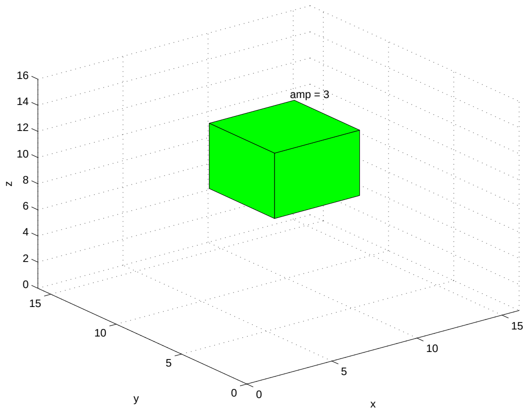
In Fig. 5.3 we can see a few steps in the feedback stage. The configuration before the Feedback is shown in (a), then in (b) we see the frame around the large anomaly. We ran the search procedure again on this region. This process was repeated for each of the other found anomalies. The final area found after the feedback appears in (c).

5.2 Two Anomalies with the Same Size and Amplitude

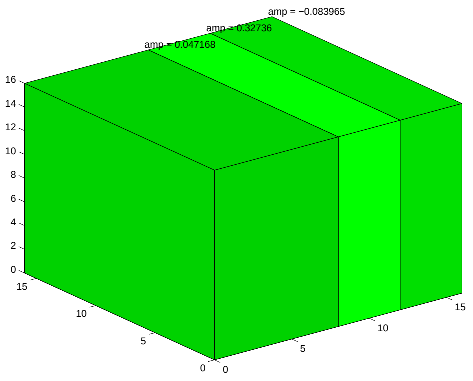
Next we considered an anomaly configuration consisting of two anomalies with the same size and amplitude. Within the total volume of 16^3 voxels, we placed two anomalies of size 4^3, one located at $(\langle 3,3,3\rangle, \langle 6,6,6\rangle)$ and the second located at $(\langle 10,10,10\rangle,\ \langle 13,13,13\rangle)$. These locations are demonstrated in Fig. 5.4(a). For this anomaly configuration we tested total SNRs between -30 and 20dB.[1]

In these experiments too, the empirical P_f level was always between 0 and $2 \cdot 10^{-4}$. The resulting P_d values for each of the anomalies are plotted in Fig. 5.4(b) against the anomaly SNR. As we expect, the results for the two anomalies are rather similar, and for both of them we have $P_d \approx 1$ for SNRs as low as -10dB. There is a slight difference between the P_f of the two anomalies

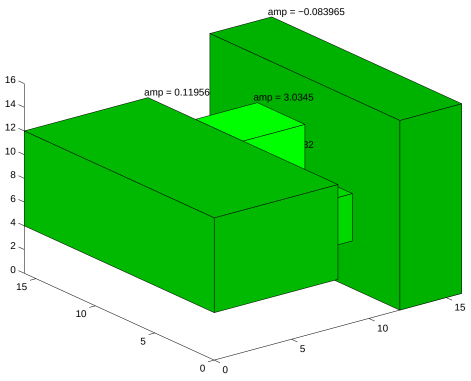
[1]Recall that the anomaly SNR is defined as the SNR of a configuration consisting only of this anomaly.



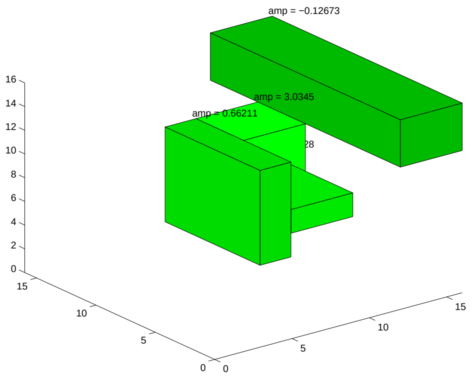
(a) Input configuration



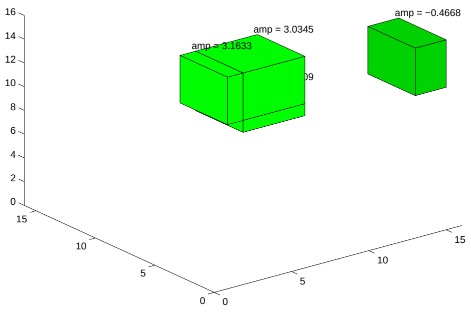
(b) Initial partition



(c) Located one anomaly

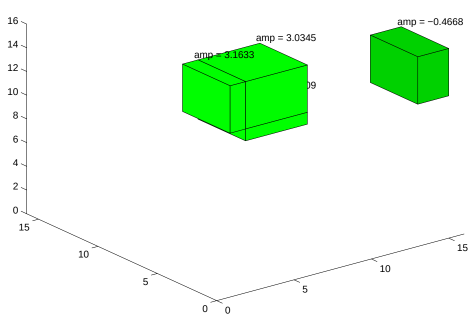


(d) Pruned some of the area

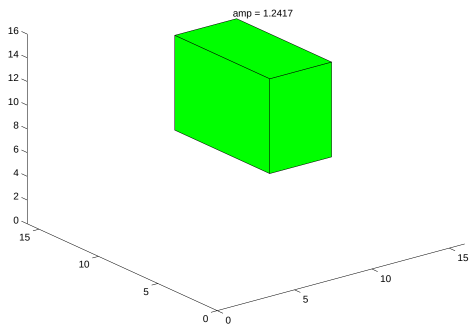


(e) Results before feedback

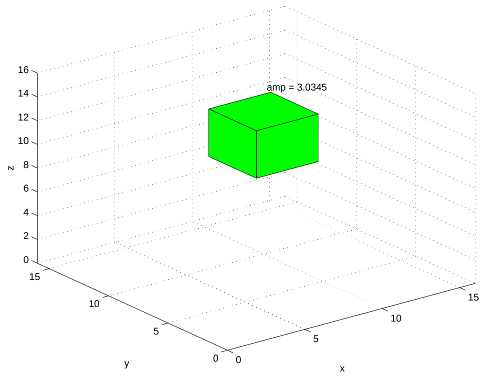
Figure 5.2: A single anomaly: snapshots of execution



(a) Before adding feedback frame

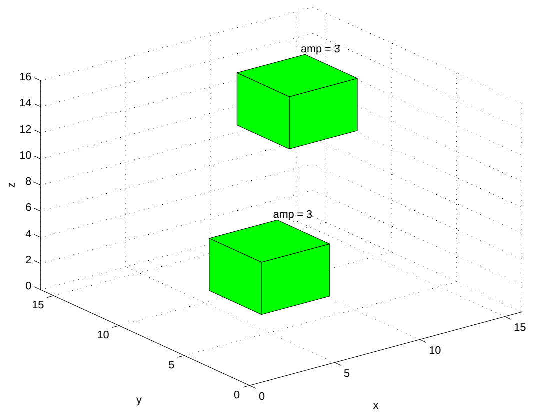


(b) After adding the frame

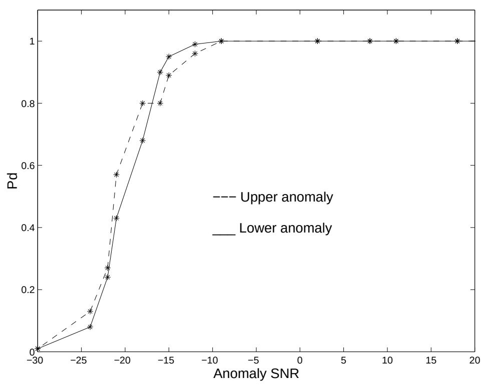


(c) After the feedback

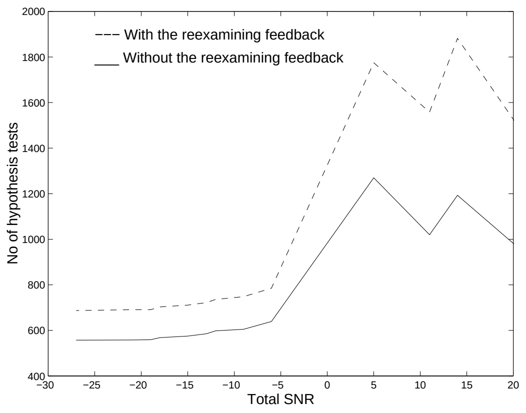
Figure 5.3: A single anomaly: snapshots of feedback



(a) Input configuration



(b) P_d vs. anomaly SNR



(c) Complexity vs. total SNR

Figure 5.4: Two anomalies with the same size and amplitude

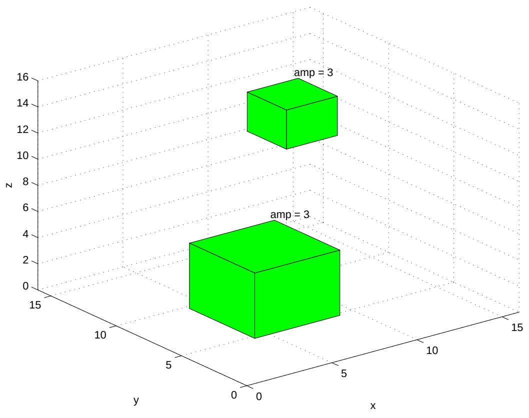
for the SNRs under -15 dB. This difference is due to the fact that the anomalies are not symmetrical relatively to the hypotheses, and therefore for extremely low SNR the second anomaly is easier to detect then the first one.

In Fig. 5.4(c) we plot the hypothesis testing complexity in this configuration against the total SNR level. It is remarkable to note that the complexity of finding two anomalies is essentially the same as the complexity of finding just one anomaly. The only thing that increases slightly is the complexity of the reexamining feedback: for two anomalies, this feedback increases the complexity of the algorithm by up to 60% (as opposed to 50% for the case of one anomaly).

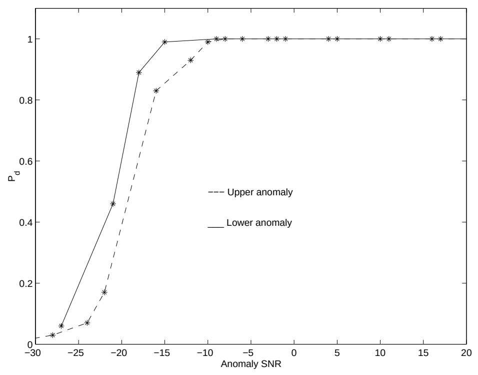
5.3 Two Anomalies of Different Sizes

This test was designed to demonstrate the capability to detect and localize two anomalies with sizes of different magnitude. The configuration consists of two anomalies with equal amplitude, but where the first anomaly size is 5^3 and the second one size is 3^3. Thus, we have the relation $SNR_1 = SNR_2 + 6.6$. The locations of these two anomalies are $(\langle 2,2,2\rangle, \langle 6,6,6\rangle)$ and $(\langle 10,10,10\rangle, \langle 12,12,12\rangle)$. This configuration is shown in Fig. 5.5(a).

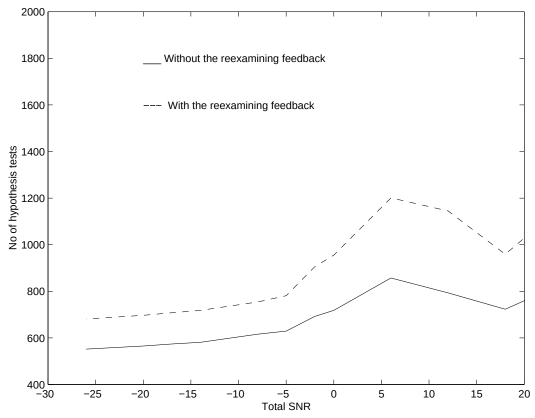
We tested total SNR levels from -30dB to 20dB. The P_f level in all these experiments was between 0 and $3 \cdot 10^{-4}$. The P_d values for the two anomalies are plotted in Fig. 5.5(b) against the anomaly SNRs. We can see that for the same levels of anomaly SNR, the P_d values are fairly close. This means that the effect of masking the small anomaly by the large one if quite small in this case. We can see that for the large anomaly we have $P_d \approx 1$ for SNR as low as -15dB and for the small anomaly we have $P_d \approx 1$ for SNR as low as -9dB. The hypothesis testing complexity for this experiment was similar to



(a) Input configuration



(b) P_d vs. anomaly SNR



(c) Complexity vs. total SNR

Figure 5.5: Two anomalies of different sizes

the complexity of the previous two experiments at the same total SNR levels.

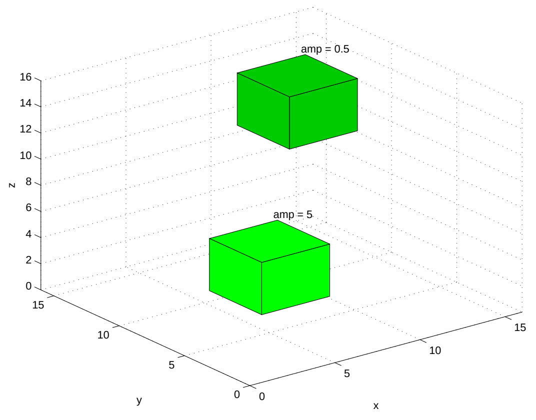
5.4 Two Anomalies of Different Amplitudes

This test was designed to demonstrate the ability of the program to detect and localize two anomalies of different amplitudes. The configuration for this test consists of two anomalies of the same size, one with amplitude ten times larger then the other one. This produces the relation $SNR_1 = SNR_2 +$ 20dB. The positions and sizes of the anomalies where chosen as in the case of the two anomalies with the same size and magnitude, i.e., The locations of these two anomalies are $(\langle 3,3,3\rangle, \langle 6,6,6\rangle)$ and $(\langle 10,10,10\rangle, \langle 13,13,13\rangle)$. This configuration is shown in Fig. 5.6(a).

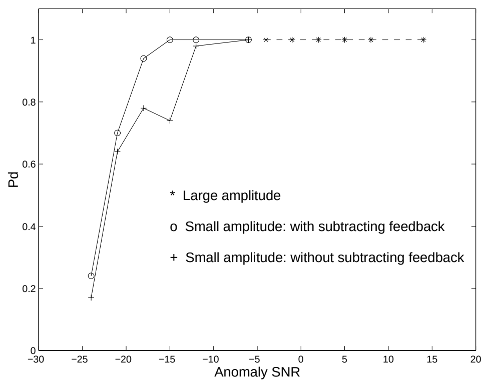
We tested total SNR levels from -5dB to 20dB. The P_f level in all these experiments was between 0 and $3 \cdot 10^{-4}$. The P_d values for the two anomalies are plotted in Fig. 5.6(b) against the anomaly SNRs.

Since the difference in the SNR of the anomalies is so large, we used this experiment to test the effectiveness of the subtracting feedback to overcome the effect of the small anomaly being masked by the large one. To this end we tested the program with and without the subtracting feedback,and compared the results. Since even without the feedback, the smaller anomaly was almost always detected for SNR levels as low as -12dB, we conclude that in this case the masking effect takes place only for very small SNRs. It can be seen from the figure that for most SNRs, the P_d values with and without the subtracting feedback are similar. However, there is a small range of low SNRs (around -15dB) where the performance has increased significantly.

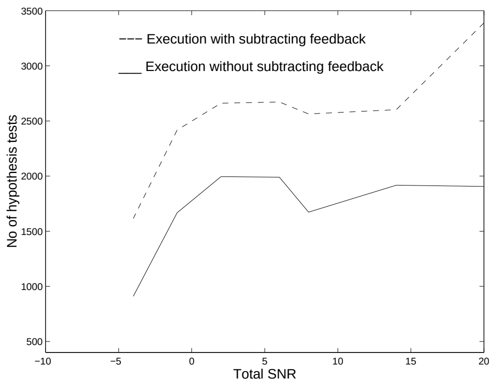
The hypothesis test complexity of these experiments is plotted in Fig. 5.6(c) against the total SNR. As we can see, the subtracting feedback increases the



(a) Input configuration



(b) P_d vs. anomaly SNR



(c) Complexity vs. total SNR

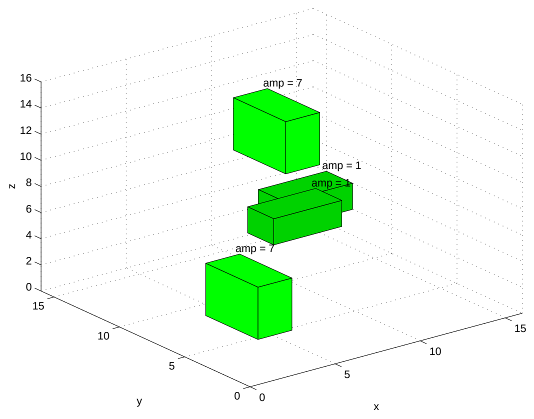
Figure 5.6: Two anomalies of different amplitudes

hypothesis testing complexity by up to 50%. Therefore, if we expect to have a small number of anomalies, the subtracting feedback is costly relative to the improvement in the probability of detection. We can also see that the complexity of these tests (even without the subtracting feedback) is larger than the case of a single anomaly by up to 30%. The reason is that the small anomaly is too large to ignore, and yet too small to be able to quickly identify. Thus, the algorithm has to keep relatively large number of regions through out the run.

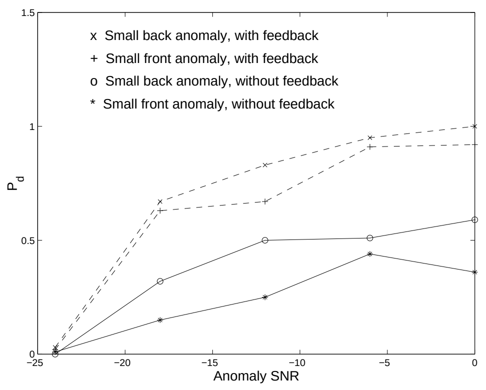
5.5 Experiments with More Anomalies

In addition to the extensive tests above, we also did some testing to explore the behavior of the algorithm when more anomalies exist. Below we show examples of configurations with three and four anomalies. These examples also demonstrate the effectiveness of the feedback mechanisms.

In the first configuration we have four anomalies, two large and two small. This configuration is depicted in Fig. 5.7(a), in which the two small anomalies appear in the middle and the two large ones appear above and below them. We tested this configurations in various SNR levels and always kept the SNR of the large anomalies 20dB above that of the small ones. The probability of detection for the large anomalies was always 1 in the examples that we tested. The probability of detection for the small anomalies with and without the subtracting feedback is described in Fig. 5.7(b). It can be seen that for this anomaly configuration the subtracting feedback is very effective. In particular, for anomaly SNRs above -5dB, the subtracting feedback increases the probability of detection from about 0.5 to 1. The number of hypothesis tests in this case was maximum 1500 without the subtracting feedback and 2400



(a) Input configuration



(b) P_d vs. anomaly SNR

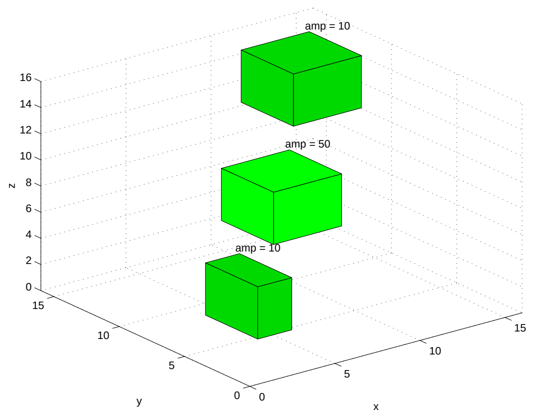
Figure 5.7: Example with four anomalies

with the reexamining feedback.

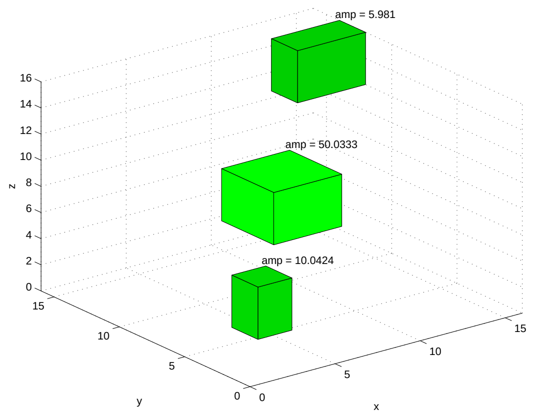
In the second example we have a configuration of three anomalies as appear in Fig. 5.8(a). The middle anomaly has SNR of about 15dB and the other two have SNR of about 0dB. As can be seen from the example, all the anomalies are detected before the reexamining feedback. However, as can be seen by comparing Fig. 5.8(b) and Fig. 5.8(c), the localization of the two small regions is greatly improved by the reexamination feedback. We remark that in this case the complexity of the reexamining feedback was about 20% of that of the main algorithm.

5.6 Complexity

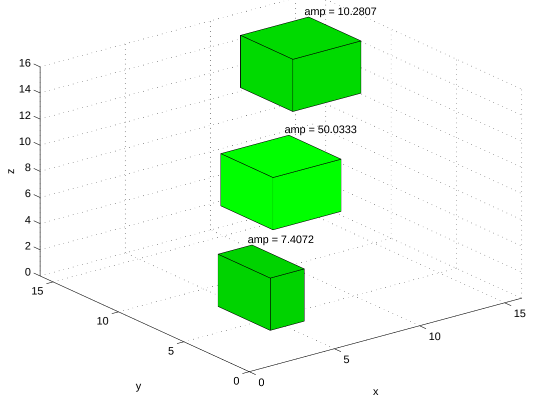
We examined the hypothesis-testing complexity of the algorithm as a function of the region size. Recall from Chapter 4 that we expect the hypothesis-testing complexity to be logarithmic in the size of the region. To test this, we ran the algorithm on regions of sizes ranging from 4^3 to 16^3. More precisely, we tried



(a) Input configuration



(b) Results before feedback



(c) Results after the feedback

Figure 5.8: Example with 3 anomalies

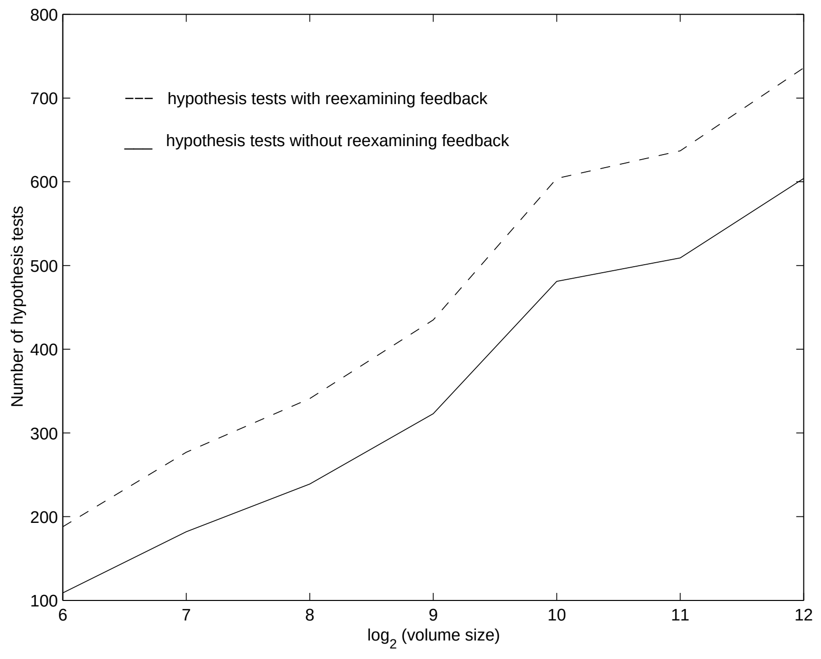


Figure 5.9: Hypothesis-testing complexity vs. region-size

the following sizes:

$$4 \times 4 \times 4, \quad 4 \times 4 \times 8, \quad 4 \times 8 \times 8, \quad 8 \times 8 \times 8,$$
$$8 \times 8 \times 16, \quad 8 \times 16 \times 16, \quad 16 \times 16 \times 16$$

In each of these regions we embedded a single anomaly of size $2 \times 2 \times 2$ in the middle of the region, and for each region size we set the amplitude of this anomaly so as to get SNR level of 5dB. For each data point we ran one hundred tests, and took the average of the hypothesis-testing complexity of all these tests. The results of this experiment are summarized in Fig. 5.9. Indeed, it can be seen in this figure that the complexity grows as a logarithmic function of the size of the region.

It can also be seen from the figure that the overhead due to the reexamining feedback is almost constant, regardless of the region size. This is also expected, since the complexity of this feedback depends only on the size of the found region before the feedback. Since we used the same anomaly size in all these tests, then the size of the found anomaly before the feedback was more or less the same in all of them, and hence the complexity of the feedback remains almost constant.

Chapter 6

Conclusions

In this thesis we developed, implemented and tested an algorithm for detection and localization of anomalies in a 3D volume, in the presence of additive Gaussian noise. This algorithm extends the scale recursive approach of Miller and Willsky [5] to the three-dimensional case. We implemented in the algorithm a mechanism due to Miller [4] to incorporate prior knowledge about the anomalies into the search via penalty functions. We also examined a few feedback methods to improve the performance of the algorithm.

We tested our algorithm on synthetically generated data consisting of a few anomaly configurations in various SNR levels. We tested the cases of a single anomaly and multiple anomalies. In all our tests, the algorithm maintained low probabilities of false-alarm (in the order of 10^{-4}), while achieving detection probabilities very close to 1, for SNR levels as low as -15dB. Also, our partial analysis and experimental data show that the complexity of the algorithm is logarithmic in the volume of the region, and our tests show that it remains quite low even for volumes of up to 16^3 voxels. We therefore conclude that this algorithm offers an inexpensive way to detect anomalies in a noisy data with very high accuracy.

We note, however, that using this algorithm in a particular application may require some modifications. In particular, the penalty functions which we used in our implementation reflect the expectations that the structure contains only a small number of anomalies, and that the size of these anomalies is relatively small. In applications where this is not the case, these penalty functions need to be adjusted. We also note that in some cases there is a problem with adjusting the penalty functions for more anomalies: In particular, in cases where the size or location of the anomaly does not match exactly any possible hypothesis of the algorithm (e.g., when the size of the hypothesis is not a power of two, see discussion in Section 5.1), the algorithm may identify parts of the anomaly as different anomalies, thus skewing the count of the anomalies.

6.1 Future Work

Some future work that can be done to better understand the effectiveness of the approach presented in this thesis include

- The algorithm should be tested with data from "real life" applications, for more realistic problems than with generated data.
- It should be possible to further reduce the complexity of the algorithm by using more sophisticated data structures to hold the results of the hypotheses tests. For example, once we determined that for a specific region on the current list, the likelihood of horizontal splitting is higher than that of vertical splitting, we never need to check the likelihood of vertical splitting of this region again.
- It may be interesting to examine the cases where there is more a-priori information about the anomalies. For example, if the exact size/shape of

the anomaly is known, then it may be possible to tailor the hypotheses or the penalty functions to improve the performance of the algorithm.

- The feedback mechanisms should be explored further. It is possible, for example, to refine the reexamining feedback so that it uses a more sophisticated method to compute the size of the frame around each anomaly (rather than always make the frame twice as large as the regions before feedback).

- Check the bound M which is the maximum number of regions the algorithm can keep at the same time. The relationship between this constant, the actual number of regions in the volume and the probability of detection for all of these regions should be explored further.

- In our tests (an also, in our complexity analysis) it was not clear what is the effectiveness of the pruning stages. Therefore, a subject to further research is to eliminate the pruning stages and examine the performance of the modified algorithm.

- The algorithm can be extended to work in more general framework, as explained in Section 3.8. Such generalizations include

  1. Non-zero normal activity. If the normal activity of the medium is not zero, but some other value $\hat{g}$. If the value $\hat{g}$ is known, then it can be subtracted from the data before further processing, and then the simple model can be used. Otherwise, further work may be required to estimate $\hat{g}$.

  2. Different noise models. If the noise is not white, not Gaussian or a correlated noise, the detection and estimation procedures, including

the MAP hypothesis testing and the maximum likelihood estimation of the amplitude should be changed according to the new noise model.

3. General linear model: In this model we view the measured data as: $y = Tg + n$, where T is a some linear transformation.

4. Non-linear model: In this model we view the measured data as: $y = T(g) + n$, where T is any general non-linear physical transformation function.

Bibliography

[1] Stanley R. Deans. *The Radon Transform and some of it's applications.* John Wiley & Sons, 1983.

[2] Austin Frakt. Multiscale hypothesis testing with application to anomaly characterization from tomographic projections. Master's thesis, MIT, May 1996.

[3] Jeffrey J. Daniels Leon Peters Jr. and Jonathan D. Young. Ground penetrating radar as a subsurface environmental sensing tool. *Proc. IEEE*, 82(12):1802–1822, December 1994.

[4] Eric L. Miller. Statically based methods for anomaly characterization in images from observations of scattered radiation. *Northeastern University, Boston MA, CDSP Center Report TR-CDSP-96-3*, January 1996.

[5] Eric L. Miller and Alan S. Willsky. Multiscale, statistical anomaly detection analysis algorithms for linearized inverse scattering problems. *Multidimensional Systems and Signal Processing*, 8, 1996.

[6] J. Neyman and E. Pearson. On the problem of the most tests of statistical hypotheses. *Philosophical Trans. of the Royal Society of London*, A231(9):289–337, 1933.

[7] Richard L. Medina Richard A. Albanese and John W. Penn. Mathematics, medicine and microwaves. *Inverse Problems*, 10:995–107, 1994.

[8] Kevin Riley and Anthony J. Devaney. Wavelet processing of images for target detection. *International Journal of Imaging Systems and Technology*, 7:404–420, 1996.

[9] David J. Rossi and Alan S. Willsky. Reconstruction from projections based on detection and estimation of objects - parts i and ii. *IEEE Trans. on Acoustics, Speech and Signal Processing*, ASSP-32, No. 4:886–906, August 1984.

[10] Charles E. Leiserson Thomas H. Cormen and Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[11] Carlos Torres-Verdin and Tarek M. Habashy. Rapid 2.5-d forward modeling and inversion via a new nonlinear scattering approximation. *Radio Sci.*, 29(4):1051–1079, July-August 1994.

[12] H.L. van Trees. *Detection Estimation and Modulation Theory.* New York: John Wiley, 1968.

Appendix A

Penalties are Necessary

Below we analyze the behavior of our hypothesis testing in the absence of penalty values, and demonstrate the tendency of the program to over-fit the data. Specifically, for the set of hypotheses in our algorithm, we compare the likelihood of the hypotheses which correspond to keeping the entire current region, taking only one of the halves of this region, or taking both halves of this region. (Note that these indeed cover all the hypotheses which our algorithm considers.) We show that if it weren't for the penalty values, our algorithm would have always preferred to take the two halves of the current region

Recall that we use the ML estimation for the amplitude in the current region (for the additive Gaussian noise model), namely

$$\hat{\mathbf{a}}(\mathbf{y}) = \left(\mathbf{B}^T\mathbf{R}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{R}^{-1}\mathbf{y}$$

where $\mathbf{R}$ is the noise covariance matrix. Below we assume that the noise is white with uniform variance, namely $\mathbf{R} = \sigma^2 I$. Therefore, we get

$$\hat{\mathbf{a}}(\mathbf{y}) = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{y} \tag{A.1}$$

Also recall that we use the MAP decision rule for the hypothesis testing. Namely we choose the hypothesis H_m where

$$m = \mathrm{argmin}_j \frac{1}{\sigma^2}(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j)^T(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j) + \pi_j \tag{A.2}$$

where $\mathbf{B}_j, \hat{\mathbf{a}}_j$, respectively, are the indicator matrix and estimated amplitude vector which describe hypothesis H_j (recall that hypothesis H_j is described formally by $\mathbf{y} = \mathbf{B}_j\hat{\mathbf{a}}_j + \mathbf{n}$).

As we can see, if we have no penalties (i.e., if $\pi_j = 0$ for all j), then this decision rule always chooses the hypothesis H_j which minimizes $(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j)^T(\mathbf{y}-\mathbf{B}_j\hat{\mathbf{a}}_j)$. Below we denote $s_j \stackrel{\mathrm{def}}{=} \|\mathbf{y}-\mathbf{B}_j\hat{\mathbf{a}}_j\|^2 = (\mathbf{y}-\mathbf{B}_j\hat{\mathbf{a}}_j)^T(\mathbf{y}-\mathbf{B}_j\mathbf{a}_j)$. Expanding this expression yields

$$s_j = (\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j)^T(\mathbf{y} - \mathbf{B}_j\hat{\mathbf{a}}_j) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{B}_j\hat{\mathbf{a}}_j - (\mathbf{B}_j\hat{\mathbf{a}}_j)^T\mathbf{y} + (\mathbf{B}_j\hat{\mathbf{a}}_j)^T(\mathbf{B}_j\hat{\mathbf{a}}_j)$$

Substituting the value of $\hat{\mathbf{a}}$ from Eq. A.1, we get

$$\begin{aligned} s_j \;=\; & \mathbf{y}^T\mathbf{y} \;-\; \mathbf{y}^T\mathbf{B}_j(\mathbf{B}_j^T\mathbf{B}_j)^{-1}\mathbf{B}_j^T\mathbf{y} \\ & -\, \mathbf{y}^T\mathbf{B}_j\left((\mathbf{B}_j^T\mathbf{B}_j)^{-1}\right)^T\mathbf{B}_j^T\mathbf{y} \;+\; \mathbf{y}^T\left((\mathbf{B}_j^T\mathbf{B}_j)^{-1}\right)^T\mathbf{B}_j^T\mathbf{B}_j(\mathbf{B}_j^T\mathbf{B}_j)^{-1}\mathbf{B}_j^T\mathbf{y} \end{aligned} \tag{A.3}$$

We now note that in our hypotheses, the different anomalous regions never overlap. This implies that $\mathbf{B}_j^T\mathbf{B}_j$ is always a diagonal matrix and therefore for all j we have $\mathbf{B}_j^T(\mathbf{B}_j^T\mathbf{B}_j)^{-1}\mathbf{B}_j = \mathbf{I}$ and $(\mathbf{B}_j^T\mathbf{B}_j)^{-1} = \left((\mathbf{B}_j^T\mathbf{B}_j)^{-1}\right)^T$. Plugging these two equations into Eq. A.3 we obtain

$$s_j = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{B}_j(\mathbf{B}_j^T\mathbf{B}_j)^{-1}\mathbf{B}_j^T\mathbf{y} \tag{A.4}$$

A.1 Comparing H_0 and H_1

We proceed now to compare between the likelihoods of hypotheses H_0 and H_1 (which are depicted in Fig. A.1). The $\mathbf{B}$ matrices which correspond to these

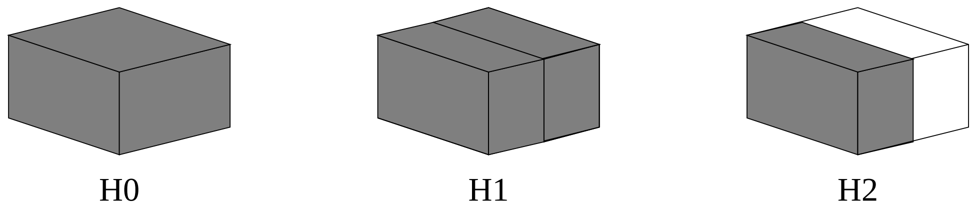


Figure A.1: Localization hypotheses H_0, H_1 and H_2

hypotheses are

$$\mathbf{B}_0 = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 \end{pmatrix}^T \text{ and } \mathbf{B}_1 = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 \end{pmatrix}^T$$

so we can write $\mathbf{B}_0 = \mathbf{B}_1 \cdot [1\ 1]^T$. Using Eq. A.4, we get

$$s_1 - s_0 = \mathbf{y}^T \mathbf{B}_0 (\mathbf{B}_0^T \mathbf{B}_0)^{-1} \mathbf{B}_0^T \mathbf{y} - \mathbf{y}^T \mathbf{B}_1 (\mathbf{B}_1^T \mathbf{B}_1)^{-1} \mathbf{B}_1^T \mathbf{y} \tag{A.5}$$

Denote now the number of voxels in the whole volume by N, and also denote the number of voxels in the two "anomalous regions" according to hypothesis H_1 by N_1, N_2 respectively. Then we have $N_1 + N_2 = N$. (We remark that in our case we actually have $N_1 = N_2 = \frac{N}{2}$, but the analysis below also works when this is not the case.) Using these notation we can write

$$(\mathbf{B}_0^T \mathbf{B}_0)^{-1} = \frac{1}{N} \text{ and } (\mathbf{B}_1^T \mathbf{B}_1)^{-1} = \begin{pmatrix} \frac{1}{N_1} & 0 \\ 0 & \frac{1}{N_2} \end{pmatrix}$$

and since $\mathbf{B}_0 = \mathbf{B}_1 \cdot [1\ 1]^T$, then

$$\mathbf{B}_0 (\mathbf{B}_0^T \mathbf{B}_0)^{-1} \mathbf{B}_0^T = \mathbf{B}_1 \begin{pmatrix} \frac{1}{N} & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} \end{pmatrix} \mathbf{B}_1^T$$

Let us now define the matrix Z as

$$Z \overset{\text{def}}{=} \begin{pmatrix} \frac{1}{N} & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} \end{pmatrix} - (\mathbf{B}_1^T \mathbf{B}_1)^{-1} = \frac{1}{N} \begin{pmatrix} -\frac{N_2}{N_1} & 1 \\ 1 & -\frac{N_1}{N_2} \end{pmatrix}$$

It can be easily verified that Z is a negative-semi definite matrix. Finally, we obtain that

$$s_1 - s_0 = \mathbf{y}^T \mathbf{B}_1 Z \mathbf{B}_1^T \mathbf{y} \leq 0$$

Therefore, without priors, the algorithm would always prefer to divide a volume then to keep it as a whole. We note that this property holds for any partition of the volume, even if the two parts are not of equal size. Therefore, to keep the data from over-fitting, we need to have penalties for area division.

A.2 Comparing H_1 and H_2

Below we compare the likelihood of the hypotheses H_1 and H_2. The $\mathbf{B}$ matrices for these hypotheses are:

$$\mathbf{B}_1 = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 \end{pmatrix}^T \text{ and } \mathbf{B}_2 = \begin{pmatrix} 1 & \dots & 1 & 0 & \dots & 0 \end{pmatrix}^T$$

so we can write $\mathbf{B}_2 = \mathbf{B}_1 \cdot [1\ 0]^T$. As before, we denote the number of voxels in the whole volume by N, the number of voxels in the upper part by N_1 and the number of voxels in the lower part by N_2 (so that $N = N_1 + N_2$). Here we have

$$(\mathbf{B}_2^T \mathbf{B}_2)^{-1} = \frac{1}{N_1} \text{ and } (\mathbf{B}_1^T \mathbf{B}_1)^{-1} = \begin{pmatrix} \frac{1}{N_1} & 0 \\ 0 & \frac{1}{N_2} \end{pmatrix}$$

Since $\mathbf{B}_2 = \mathbf{B}_1 \cdot [1\ 0]^T$, we get

$$\mathbf{B}_2 (\mathbf{B}_2^T \mathbf{B}_2)^{-1} \mathbf{B}_2^T = \mathbf{B}_1 \begin{pmatrix} \frac{1}{N_1} & 0 \\ 0 & 0 \end{pmatrix} \mathbf{B}_1^T$$

If we now define the matrix

$$Z \overset{\text{def}}{=} \begin{pmatrix} \frac{1}{N_1} & 0 \\ 0 & 0 \end{pmatrix} - (\mathbf{B}_1^T \mathbf{B}_1)^{-1} = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{1}{N_2} \end{pmatrix}$$

then again Z is negative semi-definite, and we have

$$s_1 - s_2 = \mathbf{y}^T\mathbf{B}_2(\mathbf{B}_2^T\mathbf{B}_2)^{-1}\mathbf{B}_2^T\mathbf{y} - \mathbf{y}^T\mathbf{B}_1(\mathbf{B}_1^T\mathbf{B}_1)^{-1}\mathbf{B}_1^T\mathbf{y} \;=\; \mathbf{y}^T\mathbf{B}_1 Z \mathbf{B}_1^T\mathbf{y} \leq 0 \tag{A.6}$$

Therefore, if we have no priors then when dividing a region, the hypothesis which keeps both parts will always be preferred to the hypothesis which keeps only one part. As before, we note that this analysis does not assume that the two regions in the partition are of the same size.

A.3 More Than one Region

Below we show that the above results hold even when the current configuration contains more than one region. Intuitively, this is true since all the regions are disjoint and thus they do not interfere with each other. Formally, suppose that we have k regions in the current configuration, namely

$$\mathbf{B} = (\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_k)$$

In the localization step we consider the following $12k + 1$ hypotheses:

- H_0 (the null hypothesis): This hypothesis keeps all the regions unchanged
- $H_{j,i}$ ($j = 1, \dots, 12, \ i = 1, \dots, k$): Apply the partition from hypothesis H_j in Section 3.2 to the i'th region, keeping the other regions unchanged.

As before, in our analysis we only consider the hypotheses $H_0, H_{1,i}$ and $H_{2,i}$. To simplify the notation in the discussion below, we sometime write $H_{0,i}$ (with $i = 1, 2, \dots, k$) instead of H_0. The $\mathbf{B}$ matrices which represent these hypotheses are

$$\mathbf{B}_{0,i} \;=\; \mathbf{B}$$

$$\begin{aligned} \mathbf{B}_{1,i} &= (\mathbf{b}_1 \ \ldots \ \mathbf{b}_{i-1} \ \mathbf{b}_{i,1} \ \mathbf{b}_{i,2} \ \ldots \mathbf{b}_k) \\ \mathbf{B}_{2,i} &= (\mathbf{b}_1 \ \ldots \ \mathbf{b}_{i-1} \ \mathbf{b}_{i,2} \ \ldots \mathbf{b}_k) \end{aligned}$$

where $\mathbf{b}_{i,1}, \mathbf{b}_{i,2}$ are the indicator vectors of the two sub-regions which are obtained in the partition of region i.

Below we show that if we had no priors, then for every $i = 1, \ldots, k$, hypothesis $H_{1,i}$ would always be preferred over hypotheses $H_{0,i}$ and $H_{2,i}$. Let us now denote for any $i = 1, \ldots, k$

$$s_{j,i} \overset{\text{def}}{=} \|\mathbf{y} - \mathbf{B}_{j,i}\hat{\mathbf{a}}_{j,i}\|^2 \quad (j = 0, 1, 2)$$

as before, if we had no priors then the localization would have picked the hypothesis which minimizes $s_{i,j}$. Also, similarly to Eq. A.4 we get

$$s_{j,i} = \mathbf{y}^T\mathbf{y} \ - \ \mathbf{y}^T \ \mathbf{B}_{j,i}(\mathbf{B}_{j,i}^T\mathbf{B}_{j,i})^{-1}\mathbf{B}_{j,i} \ \mathbf{y}$$

We now note that since all the regions in the current configuration are disjoint, then all the matrices $(\mathbf{B}_{j,i}^T\mathbf{B}_{j,i})^{-1}$ are diagonal, and therefore we have

$$\begin{aligned} \mathbf{y}^T\mathbf{B}_{0,i}(\mathbf{B}_{0,i}^T\mathbf{B}_{0,i})^{-1}\mathbf{B}_{0,i}^T\mathbf{y} &= \mathbf{y}^T\mathbf{b}_1(\mathbf{b}_1^T\mathbf{b}_1)^{-1}\mathbf{b}_1^T\mathbf{y} \\ &+ \ldots \\ &+ \mathbf{y}^T\mathbf{b}_k(\mathbf{b}_k^T\mathbf{b}_k)^{-1}\mathbf{b}_k^T\mathbf{y} \end{aligned}$$

and similarly

$$\begin{aligned} \mathbf{y}^T\mathbf{B}_{1,i}(\mathbf{B}_{1,i}^T\mathbf{B}_{1,i})^{-1}\mathbf{B}_{1,i}^T\mathbf{y} &= \mathbf{y}^T\mathbf{b}_1(\mathbf{b}_1^T\mathbf{b}_1)^{-1}\mathbf{b}_1^T\mathbf{y} \\ &+ \ldots \\ &+ \mathbf{y}^T\mathbf{b}_{i,1}(\mathbf{b}_{i,1}^T\mathbf{b}_{i,1})^{-1}\mathbf{b}_{i,1}^T\mathbf{y} \\ &+ \mathbf{y}^T\mathbf{b}_{i,2}(\mathbf{b}_{i,2}^T\mathbf{b}_{i,2})^{-1}\mathbf{b}_{i,2}^T\mathbf{y} \\ &+ \ldots \\ &+ \mathbf{y}^T\mathbf{b}_k(\mathbf{b}_k^T\mathbf{b}_k)^{-1}\mathbf{b}_k^T\mathbf{y} \end{aligned}$$

and

$$\begin{aligned}\mathbf{y}^T\mathbf{B}_{2,i}(\mathbf{B}_{2,i}^T\mathbf{B}_{2,i})^{-1}\mathbf{B}_{2,i}^T\mathbf{y} &= \mathbf{y}^T\mathbf{b}_1(\mathbf{b}_1^T\mathbf{b}_1)^{-1}\mathbf{b}_1^T\mathbf{y}\\ &+\dots\\ &+\mathbf{y}^T\mathbf{b}_{i,2}(\mathbf{b}_{i,2}^T\mathbf{b}_{i,2})^{-1}\mathbf{b}_{i,2}^T\mathbf{y}\\ &+\dots\\ &+\mathbf{y}^T\mathbf{b}_k(\mathbf{b}_k^T\mathbf{b}_k)^{-1}\mathbf{b}_k^T\mathbf{y}\end{aligned}$$

Thus, for any i we have

$$s_{i,1} - s_{i,0} = \mathbf{y}^T\mathbf{C}_{0,i}(\mathbf{C}_{0,i}^T\mathbf{C}_{0,i})^{-1}\mathbf{C}_{0,i}^T\mathbf{y} - \mathbf{y}^T\mathbf{C}_{1,i}(\mathbf{C}_{1,i}^T\mathbf{C}_{1,i})^{-1}\mathbf{C}_{1,i}^T\mathbf{y}$$

and

$$s_{i,1} - s_{i,2} = yy^T\mathbf{C}_{2,i}(\mathbf{C}_{2,i}^T\mathbf{C}_{2,i})^{-1}\mathbf{C}_{2,i}^T\mathbf{y} - \mathbf{y}^T\mathbf{C}_{1,i}(\mathbf{C}_{1,i}^T\mathbf{C}_{1,i})^{-1}\mathbf{C}_{1,i}^T\mathbf{y}$$

where the matrices $\mathbf{C}_{0,1}, \mathbf{C}_{1,i}, \mathbf{C}_{2,i}$ are defined

$$\mathbf{C}_{0,1} = (\mathbf{b}_i), \quad \mathbf{C}_{1,i} = (\mathbf{b}_{i,1}\ \mathbf{b}_{i,2}), \quad \text{and} \quad \mathbf{C}_{2,i} = (\mathbf{b}_{i,2})$$

We now note that these expressions are essentially the same as the expressions in Eq. A.5 and A.6, and thus the same analysis from above yields that $s_{i,1} - s_{i,0} \leq 0$ and also $s_{i,1} - s_{i,2} \leq 0$.

A.4 Conclusion

As we have seen, if we had no priors then the hypothesis of keeping both halves would be preferred over keeping the whole region, only the left half of the region or only the right half of the region. Obviously, this is true in every dimension. We also note that the analysis does not depend on the form of the noise (and not even on its existence)! We conclude that our algorithm always has a tendency to divide a volume and keep both regions if there are no penalties.

Appendix B

Complexity Analysis with Pruning

In this appendix we analyze the hypothesis testing complexity of the main search routine of our algorithm. This analysis is a joint work with Shai Halevi. Recall that the main search routine of the algorithm works by maintaining a *current list* of regions, and at each step either splitting one of the regions in this list (in a localization step) or dropping it altogether (in a pruning step). We again consider the execution tree for a particular execution, and show a bound on the number of regions in it. To obtain this bound we establish a recurrence formula that describes the worst-case behavior of the algorithm, and then show how to bound the function which is defined by this recurrence. We refer to [10, Chapter 4] for an exposition of recurrence formulas.

For any N, M, denote the maximum number of regions in an execution tree for initial region-size N and maximum list-size M by $T(N, M)$. Now fix some N, M and consider an execution which actually achieves this maximum number of nodes. Consider the first step in the algorithm: In this step, the root is split into two regions, each of size $\frac{N}{2}$. Below we refer to these regions as the left and right halves, and we refer to their sub-trees in the execution tree as the left and right sub-trees, respectively.

Without loss of generality, we assume that at the end of the algorithm there is at least one leaf which is not pruned by the algorithm. (if this is not true, we can stop one step before the last leaf was pruned, and this does not change the number of nodes in our tree). Assume, then, that this leaf belongs to the left sub-tree (the case where it belongs to the right sub-tree is symmetric). This means that throughout the algorithm, the current list contains at least one region from the left sub-tree, and therefore it contains at most $M-1$ regions from the right sub-tree. Therefore, the number of regions in the left sub-tree is at most $T(\frac{N}{2}, M-1)$. Also, the left sub-tree contains at most $T(\frac{N}{2}, M)$ regions, since in the worst case we pruned away at some point all the leaves from the right sub-tree, and where left with a bound of M regions for the left sub-tree. If we add the root to this count, we obtain the following recurrence

$$T(N, M) \leq T(\frac{N}{2}, M-1) + T(\frac{N}{2}, M) + 1$$

The border conditions for this recurrence are $T(1, M) = 1$ (because we cannot split a region of size 1), and $T(N, 1) = \log_2 N$ (since if we only keep one region on the list, then we can only have one path from root to leaf in the tree).

In fact, by slightly modifying the example from Chapter 4 it is not hard to see that a worst-case execution satisfies the above recurrence with equality: For particular values of N, M, the execution starts by splitting the root region and keeping both halves. Next the right sub-tree is explored according to a worst-case execution $T(\frac{N}{2}, M-1)$, while the left half remains untouched. Then the right sub-tree is pruned, and the left sub-tree is explored according to a worst-case execution $T(\frac{N}{2}, M)$. Indeed, for this execution we have $T(N, M) \overset{\text{def}}{=} T(\frac{N}{2}, M-1) + T(\frac{N}{2}, M) + 1$.

To solve this recurrence, it is easier to work with $n \overset{\text{def}}{=} \log_2 N$ than with N itself. We thus define the function $t(n, M) \overset{\text{def}}{=} T(2^n, M)$, and then this function satisfies the recurrence

$$t(n, M) = t(n-1, M-1) + t(n-1, M) + 1$$

with border conditions $t(0, M) = 1$ and $t(n, 1) = n$ for every M, n. Below we prove that for any n, M this last function satisfies

$$\binom{n}{M} < T(n, M) < \binom{n+M}{M}$$

We prove these bounds by induction over both M, n. It can be verified that the bounds hold for the border conditions since

$$\begin{array}{rlll} & \binom{0}{M} = 0 & < t(0, M) = 1 & \leq \binom{0+M}{M} = 1 \\ \text{and} & \binom{n}{1} = n & \leq t(n, 1) = n & < \binom{n+1}{1} = n+1 \end{array}$$

For the induction step, we use the identity $\binom{a}{b} = \binom{a-1}{b} + \binom{a-1}{b-1}$. Assume that the above bounds hold for any pair n', M' with either $n' < n$ or $M' < M$, and we prove that they also hold for the pair n, M. For the lower bound, we have

$$t(n, M) = t(n-1, M) + t(n-1, M-1) + 1 \geq \binom{n-1}{M} + \binom{n-1}{M-1} + 1 = \binom{n}{M} + 1 > \binom{n}{M}$$

and for the upper bound we have

$$\begin{aligned} t(n, M) &= t(n-1, M) + t(n-1, M-1) + 1 \\ &\leq \binom{n+M-1}{M} + \binom{n+M-2}{M-1} + 1 \\ &\leq \binom{n+M-1}{M} + \binom{n+M-1}{M-1} = \binom{n+M}{M} \end{aligned}$$

We thus conclude that the worst case execution tree contains between $\binom{\log_2 N}{M}$ and $\binom{\log_2 N+M}{M}$ regions. Since for each region in the tree we can have at most $12M + 1$ localization hypotheses and M pruning hypotheses, then we get

Theorem B.1 *The number of hypothesis-tests in every execution of the algorithm is at most* $(13M+1)\cdot\binom{\log_2 N+M}{M}$*, where* N *is the size of the input region and* M *is the maximum number of regions which are kept on the current list at the same time during this execution.*