EFFICIENT GRAPH-BASED ENERGY MINIMIZATION METHODS IN COMPUTER VISION

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Olga Veksler
August 1999

EFFICIENT GRAPH-BASED ENERGY MINIMIZATION METHODS IN COMPUTER VISION

Olga Veksler, Ph.D.
Cornell University 1999

Energy minimization is an elegant approach to computer vision. Vision problems usually have many solutions due to uncertainties in the imaging process and ambiguities in visual interpretation. The energy function encodes the problem constraints, and its minimum gives the optimal solution. Despite numerous advantages, this approach is severely limited by the high computational cost.

The main contribution of my thesis lies in developing efficient combinatorial optimization algorithms for several important classes of energy functions which incorporate everywhere smooth, piecewise constant, and piecewise smooth priors. These priors assume, respectively, that the quantity to be estimated varies smoothly over its domain, consist of several pieces with constant values, or consist of several pieces with smoothly varying values. The algorithms rely on graph cuts as a powerful optimization technique.

For a certain everywhere smooth prior we develop an algorithm which finds the exact minimum by computing a single graph cut. This method is most suitable to estimate quantities without discontinuities. But even when discontinuities exist, the method produces good results in certain cases. The running time is low order polynomial.

For several wide classes of piecewise smooth priors we develop two approximation algorithms (we show that exact minimization in NP-hard in these cases). These algorithms produce a local minimum in interesting large move spaces. Furthermore, one of them finds a solution within a known factor from the optimum. The algorithms are iterative and compute several graph cuts at each iteration. The running time at each iteration is effectively linear due to the special graph structure. In practice it takes just a few iterations to converge. Moreover most of the progress happens during the first iteration.

For a certain piecewise constant prior we adapt the algorithms developed for the piecewise smooth prior. One of them finds a solution within a factor of two from the optimum. In addition we develop a third algorithm which finds a local minimum in yet another move space.

We demonstrate the effectiveness of our approach on image restoration, stereo, and motion. For the data with ground truth, our methods significantly outperform standard methods.

Biographical Sketch

Olga Veksler was born on May 15, 1973 in Sergiev Posad, Russia. In May 1995 she received the B.A. degree from New York University with Honors in mathematics and computer science. In the fall of 1995 she entered the Ph.D. program in the Field of Computer Science at Cornell University. In January 1999 she received the M.A. degree in computer science from Cornell University.

Acknowledgements

I would like to express my graditude to my advisor Professor Ramin Zabih for his advice and guidance. During my first year at Cornell University I took a computer vision course to satisfy one of the requirements for the Ph.D. Ramin has stimulated a lot of interest in the subject, and I ended up doing research in vision. He took interest in my final project for this course, and this was the beginning of several years of fruitfull collaboration. His advise during the preparation of this dissertation was very valuable.

I am especially thankful to Yuri Boykov whom I dragged into computer vision and with whom I worked in this field ever since. During our constant discussions and arguments a lot of interesting ideas were born. His scrupulous attention to details saved us from many pitfalls.

I would like to thank Professors Eva Tardos and Jon Kleinberg for sharing their expertise in the field of combinatorial optimization and graph methods. I am especially grateful to Eva Tardos for always being available to discuss our results and providing interesting suggestions. I am very grateful to Jon Kleinberg for helping to prove some NP hardness results and for designing an algorithm for speeding up computation of max flow on similar graphs.

I would like to express my gratitude to Professor Daniel Huttenlocher for helping to create a stimulating vision environment at our department.

I would like to thank to my commitee member Professor Elizabeth Slate for insightful questions and patience while reading my thesis.

I am grateful to Y. Ohta and Y. Nakamura for supplying the ground truth imagery from the University of Tsukuba Multiview Image Database. This research has been supported by DARPA under contract DAAL01-97-K-0104, by a grant from Microsoft, and by NSF Research Infrastructure award CDA-9703470.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 The optimization approach to computer vision

The goal of computer vision is to infer information about the real world from its representation through images. This is quite a challenging task. A problem in computer vision usually has many possible solutions, due to uncertainties in the imaging process and ambiguities in visual interpretation. Some mechanism is required to evaluate the options and select a solution.

The optimization approach provides an elegant and expressive framework for discriminating among solutions. The approach consists of two major steps. First an *objective* function is formulated. This is a function from the set of all possible solutions to real numbers, and it measures the goodness of a solution. To design the objective function for a problem, it is first necessary to come up with a set of constraints that an acceptable solution should satisfy. These constraints are then formalized in the objective function so that it assigns high goodness to the solutions that satisfy the constraints well. For example the two constraints commonly used in vision are provided by the data and prior knowledge. The data constraint restricts a desired solution to be close to the observed data, and the prior constraint confines the desired solution to have the form agreeable with the prior knowledge. We are going to assume that smaller values of the objective function mean higher goodness. Thus the smaller is the value of the objective function, the better the solution, and a global minimum[1] of an objective function gives an optimal solution to the problem. All objective functions in this work are called *energy* functions.

The second step of the approach is to minimize the energy function. Although the design of a good energy function is not trivial, its optimization is even harder. Most of the interesting energy functions are not convex, i.e. they have multiple

[1]Intuitively it is desirable to design an objective function with only one global minimum. However it is not always easy to guarantee this property.

local minima. The computational demands are quite formidable, and most methods seek an approximate answer. Often one makes a compromise in the design of the energy function to make the optimization task easier. But even with the compromise the majority of the optimization schemes do not produce the exact answer.

It is equally important to design an energy function which properly encodes the constraints of a problem and to find a good minimization technique. Failure in either step will almost surely result in the failure of the algorithm. However when the optimization technique gives only an approximate solution, it is hard to determine whether the algorithm fails due to a bad choice of the energy function or because the minimization algorithm produced an answer far from the optimum.

In this work we will assume that the set of all possible solutions is finite. Usually this is not a major limitation, since the quantities to be estimated can only be computed with limited precision. When the set of all possible solutions is countable, the optimization problem is said to be combinatorial. Thus the methods we develop are combinatorial optimization methods.

There are many advantages to the optimization based approach to vision. First of all it gives a common framework for many vision problems. This allows abstracting from the details of each particular problem after an energy function is formulated. That is, once a suitable energy function for a problem is found, the standard optimization machinery can be applied to solve it.

Secondly the approach provides a clean way to formalize the constraints of the problem to be solved, and allows to encode the desired global properties in the energy function. Thirdly the energy function gives a way to measure a goodness of a solution and could be used as a guide in the minimization algorithm. Lastly the optimization approach is appealing because minimization of many energy functions can be justified using Bayesian statistics [20] or mechanics [8].

The main disadvantage of the approach is its high computational cost. For many interesting problems finding the exact minimum is NP-hard, and so there is almost no hope for a tractable method to find an exact minimum.

The main contribution of this thesis lies in providing efficient minimization methods for several interesting classes of energy functions which encode everywhere smooth, piecewise constant, and piecewise smooth priors. Although the class of energy functions we minimize is restricted, it is useful enough to be applied to a wide range of important vision problems. By restricting the class of functions we are able to come up with significantly better optimization techniques than the general-purpose minimization schemes like simulated annealing, as will be shown by the experimental data.

For an everywhere smooth prior we develop a method that finds the exact minimum of a certain energy function. For a wide class of piecewise smooth priors we develop two methods that find a local minimum in interesting move spaces (we show in the appendix that finding the exact minimum is NP-hard in these cases).

One of these algorithms produces a solution within some known factor from the optimal. This factor depends on the prior. For a piecewise constant prior we adapt the algorithms developed for the piecewise smooth prior, and one of these methods is guaranteed to be within a factor of two from the optimal solution. In addition for the piecewise constant prior we develop an algorithm that finds a local minimum in yet another interesting move space. All the methods we develop use graph cuts as a powerful optimization technique.

The energy minimization approach is very popular in the vision field, and was widely applied to many vision problems. The amount of literature on the subject is immense, see for example Horn and Schunck [23] for optical flow; Geman and Geman [20] for image restoration; Blake and Zisserman [8] for surface reconstruction, Barnard [3] for stereo matching; Derin and Elliott [15] and Hofmann et al. [22] for texture segmentation; Torre and Poggio [41] for edge detection. For a brief overview of energy minimization formulation for many vision problems see Poggio et al. [35]. We will review of several minimization methods will be given in section 1.5.

1.2 Labeling problems

Many vision problems can be formulated as labeling problems. This formulation is convenient because it gives a common notation for diverse problems. Throughout this work we will assume that a problem is posed as a labeling problem. All the examples of vision problems that we use in this work can be posed as labeling problems, as shown in Chapter 3.

To specify a labeling problem we need a set of sites and a set of labels. Sites represent image features on which we want to estimate some quantity, and labels represent the quantity to be estimated.

Let

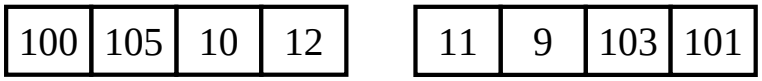
$$\mathcal{P} = \{1, 2, \ldots, n\}$$

be a set of n sites. $\mathcal{P}$ can represent pixels, edges, image segments, or other image features. $\mathcal{P}$ frequently has some natural structure, for example when it represents the set of image pixels. Pixels in an image are arranged in a two dimensional array, and each pixel which is not at the border has top, bottom, left, and right neighboring pixels. $\mathcal{P}$ can also have no natural ordering, for example when $\mathcal{P}$ represents edges. A natural ordering on the set of sites is helpful because one frequently wants to define some relationships between sites.

For all the tests performed in this work $\mathcal{P}$ represents image pixels. For this reason we are going to call $\mathcal{P}$ the set of pixels and each member of this set we are going to call a pixel.

Let

$$\mathcal{L} = \{l_1, \ldots, l_k\}$$



(a) First image (b) Second image

Figure 1.1: Example of the motion problem.

be a set of k labels. Labels represent intensities, disparities, or any other quantity to be estimated. When solving a labeling problem, one needs to measure similarity between labels. Usually a set of labels has some natural ordering which helps to design the similarity measure.

The labeling problem is to assign a label from the label set $\mathcal{L}$ to each site in the set of sites $\mathcal{P}$. Thus a labeling is a mapping from $\mathcal{P}$ to $\mathcal{L}$. We will denote a labeling by

$$f = \{f_1, \ldots, f_n\}$$

The set of all labelings $\mathcal{L}^n$ is denoted by $\mathcal{F}$.

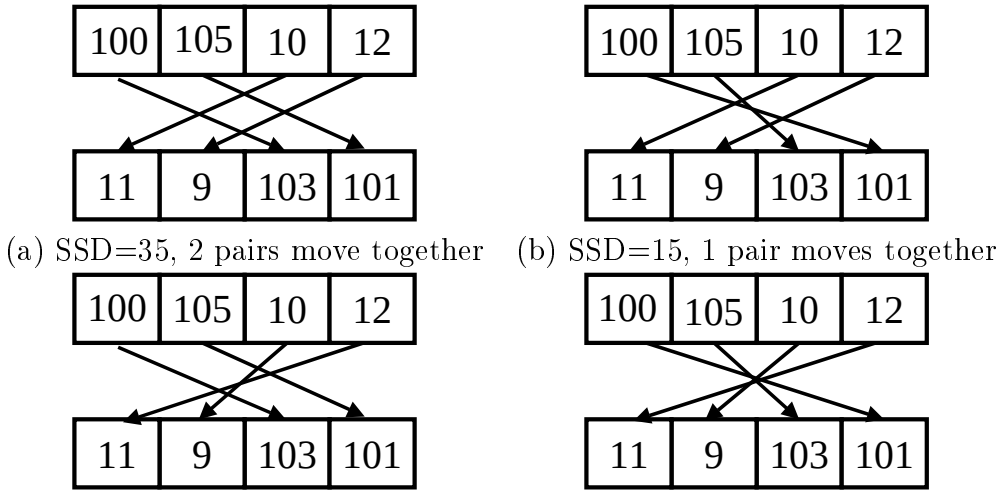
1.3 Common constraints in vision

This section describes two types of constraints commonly used in vision problems.

The most common constraint is provided by the data. Let us consider the following simple example of the motion problem. Suppose we have observed a scene with moving objects, and two observed images are shown in figure 1.1. Each pixel in the first image moves to some new pixel on the second image. These two pixels are said to correspond. The problem is to find where each pixel moves. There is some noise in the imaging system, so that the corresponding pixels do not necessarily have the same intensity. However they are expected to have similar intensities. Thus the pixel with intensity 100 in the first image could correspond to the pixel with intensity 103 or to the pixel with intensity 101 in the second image. It is very unlikely to correspond to the pixels with intensities 11 and 9. The four most plausible solutions are shown in figure 1.2. We need to choose one of them as the solution.

If we were to design an energy function based purely on the data constraints, then we might choose it to be the sum of squared differences of the corresponding pixels. The minimum of this function is achieved at the solution in figure 1.2(d).

This may be a good decision if one has no preconceived idea of what a solution should look like. However the visual world is not random; the patterns we see are structured and correlated. Therefore there exists prior knowledge which makes some labelings very likely and others highly unlikely even before the data are revealed. If prior knowledge is present, the answer given by the global minimum of the energy function which encodes only the data constraints may seem almost



(a) SSD=35, 2 pairs move together (b) SSD=15, 1 pair moves together

(c) SSD=27, 1 pair moves together (d) SSD=7, 0 pairs move together

Figure 1.2: Four most likely solutions, sums of squared differences, and number of pixel pairs which move together.

arbitrary. To get a more desirable answer, one should also encode the constraints provided by prior knowledge. This is the second type of constraint called the prior constraint.

For example in the motion problem one expects that a scene consists of objects most of which are larger than one pixel and that all pixels in the object move together. Thus we expect labelings where a lot of adjacent pixels move together. We can encode this knowledge into the energy function as negative number of pairs of pixels next to each other which move together. From the point of view of the prior knowledge, out of four labelings in figure 1.2 the one in (a) is the best.

Energy functions of the following form are very popular in vision because they express both the constraints of data and prior knowledge on the problem:

$$E(f) = E_{data}(f) + \lambda \cdot E_{prior}(f).$$

The first term $E_{data}(f)$ is called the *data energy*, and encodes the constraints of the data. $E_{prior}(f)$ is called the *prior energy*, and encodes the constraints provided by prior knowledge. The constant λ controls the relative importance of data and prior energy. The larger is λ, the more we believe in the prior information compared to the information provided by the data. For the motion example if λ is larger than 14, then solution in figure 1.2(a) has the lowest value of the energy out of all solutions in figure 1.2, and this is clearly the best solution as far as the prior knowledge is concerned. If λ is less than 8 then solution in figure 1.2(d) is optimal, and this is clearly the best solution from the point of view of the observed data.

1.4 General form of energy functions

This section describes the common form of energy functions for which the algorithms in this work are designed.

As explained in the previous section, the energy functions used in many vision applications in their most general form can be written as

$$E(f) = E_{data}(f) + \lambda \cdot E_{prior}(f). \tag{1.1}$$

The data energy E_{data} assigns heavy cost to the labelings f which do not agree with the data and small cost to the labelings close to the data. The data term is typically straight forward to design and its particular form usually depends on the noise of the imaging system. In all the methods developed in this work, the data has the form

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p).$$

Here $D_p(f_p)$ measures how much assigning label f_p to pixel p disagrees with the data. If observations at each pixel are independent, then this type of data energy is reasonable. Independence of observations at each pixel is a common approximation for the imaging process. The only restriction on $D_p(f_p)$ is that it is not negative.

The prior energy assigns heavy cost to the labelings f which are not likely from the point of view of the prior knowledge. The design of this term is more tricky and its particular form depends on the problem at hand. For example if the quantity to be estimated is a certain texture, then the prior term should assign the labeling with this texture a small penalty. If the quantity to be estimated lies on a plane, then labelings which lie close to a plane should carry small cost.

Even when one has a good idea which labelings should have high cost and which labelings should have low cost, it is frequently hard to formalize these ideas concisely. A popular choice of prior which can be easily formalized expresses smoothness constraints on the labelings. The smoothness assumption is one of the oldest in vision (see Marr and Poggio [31] or Horn and Schunck [23]). It is suitable for problems where the quantity to be estimated varies smoothly everywhere or almost everywhere. In many vision problems the quantity varies smoothly everywhere except at object boundaries where it may change abruptly. Such an abrupt change is called a *discontinuity*. In this work we deal with several kinds of smoothness priors (although some of the methods developed could be applied to other kinds of priors). For this reason we are going to rename the prior energy as the *smoothness energy* and denote it by E_{smooth}.

To formalize the smoothness energy we need to model how pixels interact with each other. To express a wide variety of smoothness energies it is enough to model how each pixel p interacts with a small set of pixels nearby. This set of pixels is called the *neighborhood* of p, and is denoted by N_p; each pixel in N_p is called

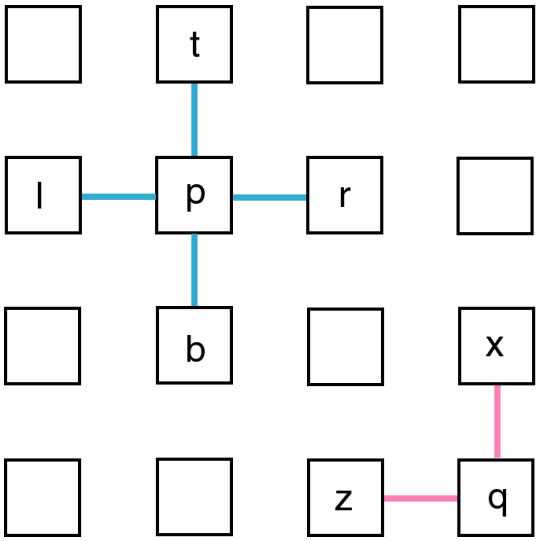


Figure 1.3: $N_p = \{t, b, l, r\}$; $N_q = \{x, z\}$.

a *neighbor* of p. For technical reasons we require N_p to satisfy the following two properties:

(a) $p \notin N_p$

(b) if $p \in N_q$ then $q \in N_p$

When $\mathcal{P}$ is rectangular image, a common structure for N_p is the so-called 4-neighbor system, shown in figure 1.3. Each pixel has the top, bottom, left, and right pixel as its neighbors. We use the 4-neighbor system for all the experiments, although our framework allows for arbitrary N_p.

Let $\mathcal{N}$ be the set of all neighboring pairs $\{p, q\}$. $\mathcal{N}$ is called the neighborhood system. Given the neighborhood system $\mathcal{N}$, the smoothness energy has the following form

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q). \tag{1.2}$$

We call $V_{\{p,q\}}(f_p, f_q)$ a *neighbor interaction* function. This function gives penalties to neighboring pixels p and q if they have different labels. Note that this penalty may depend on the particular pair of neighbors p and q and on the particular labels f_p and f_q assigned to them. The form of $V_{\{p,q\}}$ determines the type of smoothness prior, as discussed below. Thus the smoothness energy E_{smooth} is just the sum of neighbor interaction functions for all neighbor pairs. It assigns the labelings which are not smooth a high cost by counting all penalties between neighbor pairs having different labels.

Notice that we consider only pairwise interactions between pixels. This allows us to model properties which depend on the first derivative of the labeling, but will not allow us to model properties depending on the higher order derivatives.

Adding 1.1 and 1.2 we get the general form of the energy functions we are going to minimize

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q).$$

The first term in the sum expresses the data constraints and the second term expresses the smoothness constraints on the problem.

There are several major types of smoothness priors. We are going to describe the three types for which we have developed algorithms.

Everywhere smooth prior

The everywhere smooth prior assigns low cost to labelings which are smooth everywhere. For example consider figure 1.4. The labeling in (a) has low cost because labels of neighboring pixels differ by a small amount. The labeling in (b) has high cost because there is a vertical discontinuity in the middle of the image which carries a heavy penalty under the everywhere smooth prior. To formalize this prior, one chooses the neighbor interaction function $V_{\{p,q\}}(f)$ to assign higher penalties for larger differences between labels f_p and f_q. An example of such a function is shown in figure 1.5. For small values of $|f_p - f_q|$ the penalty is not large, so small variations in labels do not carry heavy penalties. However starting at some larger value of $|f_p - f_q|$ the penalty becomes too large to tolerate. As the result the optimal labeling most likely will not be allowed to have such a large drop in labels between neighboring pixels. The problem with this prior is that in most vision problems the quantity to be estimated has discontinuities, and large differences in labels should be allowed in the optimal solution. The labelings one gets using an everywhere smooth prior tend to be over-smoothed around discontinuities. However in many applications it is particularly important to recover the data around discontinuities correctly.

Piecewise constant prior

The piecewise constant prior assigns low cost to labelings which consist of one or several pieces with constant labels. For example consider figure 1.6. The labeling in (a) has low cost because it consists of only two pieces with labels 50 and 200, but the labeling in (b) has high cost because it consists of sixteen pieces with different labels. This prior can be formalized by making $V_{\{p,q\}}(f)$ zero if $f_p = f_q$ and some constant otherwise. See figure 1.7 for an example. This type of prior is more suitable than the everywhere smooth prior for problems where the quantity to be estimated has discontinuities. However it is not expressive enough for some applications.

50	52	51	49
53	49	52	50
54	50	53	52
52	49	51	53

(a) Low cost labeling

50	52	201	203
53	49	202	205
54	50	204	200
52	49	202	201

(b) High cost labeling

Figure 1.4: Examples of low cost and high cost labeling for everywhere smooth prior.

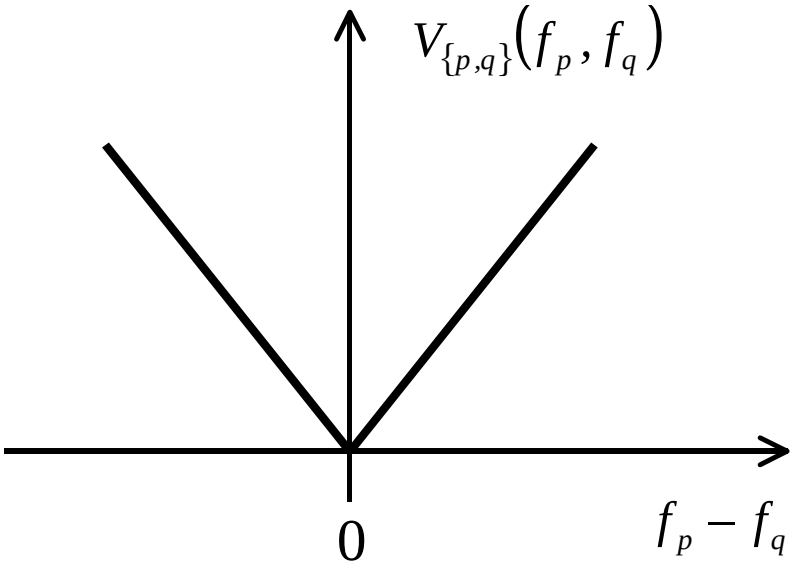


Figure 1.5: Graph of $V_{\{p,q\}}(f_p, f_q)$ for everywhere smooth prior.

50	50	200	200
50	50	200	200
50	50	200	200
50	50	200	200

(a) Low cost labeling

50	52	51	49
53	49	52	50
54	50	53	52
52	49	51	53

(b) High cost labeling

Figure 1.6: Examples of low cost and high cost labeling for piecewise constant prior.

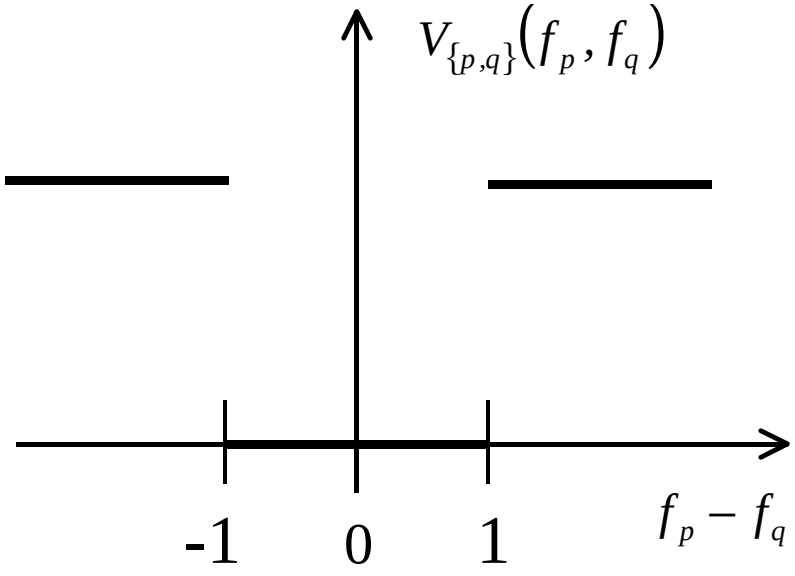


Figure 1.7: Graph of $V_{\{p,q\}}(f_p, f_q)$ for piecewise constant prior.

50	52	201	203
53	49	202	205
54	50	204	200
52	49	202	201

(a) Low cost labeling

150	82	150	49
63	249	30	250
254	98	53	95
72	149	151	53

(b) High cost labeling

Figure 1.8: Examples of low cost and high cost labeling for piecewise smooth prior.

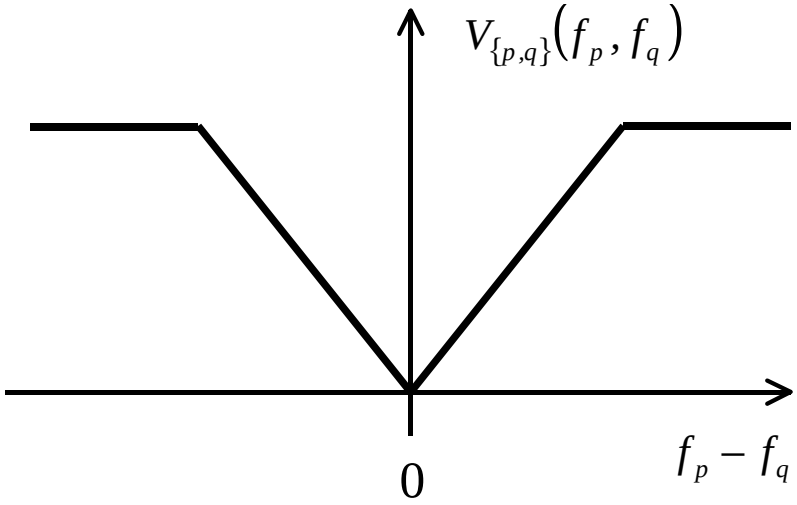


Figure 1.9: Graph of $V_{\{p,q\}}(f_p, f_q)$ for piecewise smooth prior.

Piecewise smooth prior

The piecewise smooth prior assigns low cost to labelings that consist of one or several pieces with labels varying smoothly within each piece. For example consider figure 1.8. The labeling in (a) has low cost because it consists of two pieces with labels smoothly varying around 50 in one piece and 200 in another. The labeling in (b) has high cost because it does not consist of large regions of smoothly varying labels. This prior can be formalized by choosing $V_{\{p,q\}}(f)$ which assigns a larger penalty for the larger difference between labels f_p and f_q, but sets a limit on how large this penalty can be. Consider figure 1.9 for an example. As with the graph for everywhere smooth prior there is a small penalty for small values of $|f_p - f_q|$, with larger penalties assigned to larger values of $|f_p - f_q|$. This allows labels to vary smoothly. However notice that the penalty stops growing at a certain point. This allows discontinuities to form because even if $|f_p - f_q|$ is very large, the penalty for assigning labels f_p and f_q to neighboring pixels is never too large. The piecewise smooth prior is applicable to a wider range of problems than the previous two priors.

1.5 Related work

This section describes the energy minimization methods that are most prevalent in vision. The methods can be divided into two categories: methods which search for the global minimum and methods which search for a local minimum.

1.5.1 Global energy minimization

The problem of finding the global minimum of an arbitrary energy function is intractable. Suppose there are n sites and k labels. Thus there are k^n distinct labelings. Let $E_{\hat{f}}(f)$ be an energy function which is one everywhere except at the labeling $\hat{f}$ where it is zero. There are k^n such distinct energy functions. An algorithm which can minimize all of these functions will obviously have to look at each of k^n labelings. As a consequence, any general-purpose energy minimization algorithm will require exponential time to find the global minimum.

Simulated annealing

The only general-purpose global energy minimization method in widespread use is simulated annealing. It was introduced independently by Cerny in [12] and Kirkpatrick et. al. in [28] and popularized for vision by Geman and Geman in [20]. There are several variants of this method, we will describe the version in [20]. For a comprehensive review, see [32].

The algorithm simulates the physical annealing of a material. Physical annealing is a process which determines the low energy states of a material by gradually lowering the energy. Simulated annealing is controlled by a parameter called temperature. In the beginning the temperature is set to a high value, and then lowered according to a cooling schedule. The algorithm is initialized with some labeling. Then pixels are visited in some order and at each pixel, a local random change to a labeling is made. The change is accepted if it lowers the energy function. Otherwise the change is accepted with a certain probability controlled in part by the temperature parameter. The lower the temperature, the less likely this change will be accepted. Thus at high temperature the algorithm essentially performs a random walk, and at low temperature the algorithm essentially searches for a local minimum.

With certain cooling schedules, annealing can be guaranteed to find the global minimum in the limit [20]. Not surprisingly, however, the schedules that lead to this guarantee are prohibitively slow in practice, and so sub-optimal schedules are used in practice. When annealing algorithms are used in practice, they are generally not expected to find a global minimum. They are used because they do not get stuck in one particular local minimum.

Graduated nonconvexity

An alternative to simulated annealing is to use methods that have optimality guarantees in certain cases. Continuation methods, such as GNC [8], is an example. These methods involve approximating an intractable (non-convex) energy function by a sequence of energy functions, beginning with a tractable (convex) approximation. At every step in the approximation, a local minimum is found using the solution from the previous step as the starting point. There are circumstances where these methods are known to compute the optimal solution (see [8] for details). Continuation methods can be applied to a large number of energy functions, but except for these special cases nothing is known about the quality of their output.

Mean field approach

For a given energy function $E(f)$ one can define a statistical model by

$$P(f) = Z^{-1} e^{-\frac{1}{T}E(f)}.$$

Here Z is the normalizing constant commonly called the partition function, and T is an external parameter called temperature. When temperature approaches zero, the mean quantities of the field, $\bar{f}$, minimize the energy function. Similar to simulated annealing, $\bar{f}$ is first estimated at high temperature and then tracked down using a continuation method (see Wasserstrom [42]) as the temperature gradually lowered

to zero. When the partition function Z is known, $\bar{f}$ can be deduced from it. Thus mean field theory methods concentrate on analyzing the partition function. Saddle point approximations (Parisi [33]) are used for computing the partition function. Geiger and Yuille [19] provide an interesting connection between mean field approximation and other minimization methods like graduated nonconvexity.

Dynamic programming

There are a few interesting energy functions where the global minimum can be rapidly computed, via dynamic programming or graph cuts. Dynamic programming [2] is restricted to energy functions which are essentially one-dimensional. This includes some important cases, such as snakes [27]. In general, the two-dimensional energy functions that arise in early vision cannot be solved efficiently via dynamic programming. However dynamic programming can reduce the complexity of certain two dimensional energy functions (although the resulting complexity is still exponential), see Derin and Elliot [15].

Graph cuts

Graph cuts can be used to find the global minimum for some two-dimensional energy functions. When the label set has size two, Greig et al. [21] show how to find the global minimum of a certain energy function. When the number of labels is larger, Ferrari et al. [16] develop a method optimal within a factor of two for this energy function, however the data energy they use is very restrictive. Finally Ferrari et al. [17] develop an exact method for this type of energy function using even more restrictive data energy. The algorithms developed in this work continue the idea of using graph cuts for energy minimization.

1.5.2 Local energy minimization

Due to the inefficiency of computing the global minimum, many authors have opted for a local minimum. One problem with this is that it is difficult to determine the cause of an algorithm's failures. When an algorithm gives unsatisfactory results, it may be due either to a poor choice of energy function, or to the fact that the answer is far from the global minimum. There is no obvious way to tell which of these is the problem.[2] Another issue is that local minimization techniques are naturally sensitive to the initial estimate.

[2] In the special cases where the global minimum can be rapidly computed, it is possible to separate these issues. For example, [21] points out that the global minimum of an Ising energy function is not necessarily the desired solution for image restoration. [7,21] analyze the performance of simulated annealing in cases with a known global minimum.

Continuous labels

If the energy minimization problem is phrased in smooth terms, variational methods can be applied. These methods were popularized by Horn [23]. Variational techniques use the Euler equations, which are guaranteed to hold at a local minimum (although they may also hold elsewhere). For example, in the case of optical flow, the Euler equations result in a pair of elliptic second-order partial differential equations. A number of methods have been proposed to speed up the convergence of the resulting numerical problems, including (for example) multigrid techniques [40]. To apply these algorithms to actual imagery, of course, requires discretization.

Discrete labels

Iterated conditional modes(ICM) is a greedy technique introduced by Besag in [5]. This is an iterative algorithm and it works as follows: the sites are processed sequentially, and for each site the label which gives the largest increase of the energy function is chosen. This algorithm is very sensitive to the initial labeling.

An alternative is to use discrete relaxation labeling methods; this has been done by many authors, including [13,36,39]. In relaxation labeling, combinatorial optimization is converted into real optimization with linear constraints. Then some form of gradient descent which gives the solution satisfying the constraints is used.

1.6 Contributions and prior publications

The main contribution of this thesis lies in providing efficient minimization methods for several interesting classes of energy functions which encode everywhere smooth, piecewise constant, and piecewise smooth priors.

For an everywhere smooth prior we develop a method which finds the exact minimum of a certain energy function. For a wide class of piecewise smooth priors we develop two methods which find a local minimum in interesting move spaces (finding the exact minimum in NP-hard in these cases). One of these algorithms produces a solution within a known factor from the global minimum. This factor depends on the prior. For a piecewise constant prior we adapt the algorithms developed for the piecewise smooth prior, and one of these methods is guaranteed to be within a factor of two from the optimal solution. In addition for the piecewise constant prior we develop an algorithm that finds a local minimum in yet another interesting move space. All the methods we develop use graph cuts as a powerful optimization technique.

Some of the work presented in this thesis was previously published in [9], [11], and [10].

1.7 Outline

In chapter 2 we present some technical tools, justify minimization of the energy functions in this thesis using MAP estimation of certain Markov random fields, and give some examples of vision problems. Chapter 3 presents an algorithm for an everywhere smooth prior. Chapter 4 describes several algorithms for piecewise smooth priors. Chapter 5 adapts the algorithms of chapter 4 for a piecewise constant prior, and develops an additional algorithm for a piecewise constant prior. In the appendix we show that minimizing our energy function with the piecewise constant prior is an NP-hard problem.

Chapter 2

Preliminaries

2.1 Baysian justification

This section uses Bayesian statistics to justify the energy minimization approach for the energy functions that we use. We show that minimizing such an energy function is equivalent to finding the maximum a posteriori estimate of a certain Markov random field.

2.1.1 Markov Random Fields

Markov random fields provide a convenient prior for modeling spatial interactions between pixels. Markov Random Fields were first introduced into vision by Geman and Geman [20]. For a good introduction to MRFs in vision see Li [30] or Winkler [43].

Let $\mathcal{P}$ be a set of sites, $\mathcal{L}$ a set of labels, and $\mathcal{N}$ a neighborhood system on $\mathcal{P}$ as defined in section 1.4. Let $F = F_1, \ldots, F_n$ be a set of random variables defined on $\mathcal{P}$. Each F_p takes values in the label set $\mathcal{L}$. A particular realization of the field will be denoted by $f = \{f_p | p \in P\}$, which is also called a *configuration* of the field F. As usual, $P(F_p = f_p)$ will be abbreviated by $P(f_p)$. F is said to be a Markov random field if:

(i) $P(f) > 0 \qquad \forall f \in \mathcal{F}$.
(ii) $P(f_p | f_{\mathcal{P}-\{p\}}) = P(f_p | f_{N_p})$

where $\mathcal{P} - \{p\}$ denotes set difference, f_{N_p} denotes all labels of sites in N_p, and $\mathcal{F}$ denotes the set of all possible labelings.

The first property is needed for technical reasons to ensure that the joint probability can be uniquely determined by the local conditional probabilities (see [4] for details). The second property states that a pixel is dependent directly only on its neighbors. This is a step away from independence and allows us to model spatial interactions between pixels.

MRFs are generalizations of Markov processes. While Markov processes can model one dimensional interaction between sites, MRFs model two dimensional interactions.

MRFs can be specified either by the joint distribution or by the local conditional distributions. However local conditional distributions are subject to non-trivial consistency constraints, so the first approach is most commonly used. The Hammersley-Clifford theorem [4] gives a convenient way to specify an MRF. The theorem proves the equivalence between MRFs and Gibbs random fields.

Before defining Gibbs random fields we need to define a *clique*. A set of sites is called a clique if each member of the set is a neighbor of all the other members. A Gibbs random field can be specified by the Gibbs distribution:

$$P(f) = Z^{-1} \cdot \exp\left(-\sum_{c \in \mathbf{C}} V_c(f)\right),$$

where $\mathbf{C}$ is the set of all cliques, Z is the normalizing constant, and $\{V_c(f)\}$ are functions from a labeling to real number, called the clique potential functions.

Thus to specify an MRF we need to specify the clique potential functions. Let us specify an MRF as follows. For all cliques of size larger than two the potential functions are zero, and for the cliques of size two the potential functions are specified by

$$V_c(f) = V_{\{p,q\}}(f_p, f_q),$$

where $V_{\{p,q\}}(f_p, f_q)$ were defined in section 1.4. This defines an MRF with the joint distribution:

$$P(f) = Z^{-1} \cdot \exp\left(-\sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q)\right).$$

2.1.2 Maximum a Posteriori Estimation

In general, the field F is not directly observable in the experiment. We have to estimate its realized configuration f based on an observation d, which is related to f by means of the likelihood function $P(d|f)$. The most popular way to estimate an MRF is maximum a posteriori (MAP) estimation. The MAP-MRF framework was popularized in vision by Geman and Geman [20], and has been studied by many since (see for example Besag [5], Szeliski [38]).

MAP estimation consists of maximizing the posterior probability $p(f|d)$. From the point of view of Bayes estimation, the MAP estimate minimizes the risk under the zero-one cost function.

Using Bayes rule, the posterior probability can be written as

$$p(f|d) = \frac{p(d|f)p(f)}{p(d)}$$

Thus the MAP estimate f^* is equal to

$$\arg\max_{f\in\mathcal{F}} p(d|f)p(f)$$

We now need an appropriate model for $p(d|f)$. Let d_p be the observation at pixel p and assume that $p(d|f) = \prod_{p\in\mathcal{P}} p(d_p|f_p)$. This assumption holds, for example, when the noise at each pixel is independent. Further assume that

$$p(d_p|l) = C_p \cdot \exp(-D_p(l)) \qquad \text{for} \quad l \in \mathcal{L},$$

where C_p is the normalizing constant, and D_p was defined in section 1.4. Then the likelihood can be written as

$$p(d|f) \propto \exp\left(-\sum_{p\in\mathcal{P}} D_p(f_p)\right).$$

Writing out $p(d)$ and $p(d|f)$ with the above assumptions, we get

$$f^* = \arg\max_{f\in\mathcal{F}} \exp\left(-\sum_{\{p,q\}\in\mathcal{N}} V_{\{p,q\}}(f_p, f_q) - \sum_{p\in\mathcal{P}} D_p(f_p)\right)$$

which is equivalent to minimizing

$$E(f) = \sum_{\{p,q\}\in\mathcal{N}} V_{\{p,q\}}(f_p, f_q) + \sum_{p\in\mathcal{P}} D_p(f_p),$$

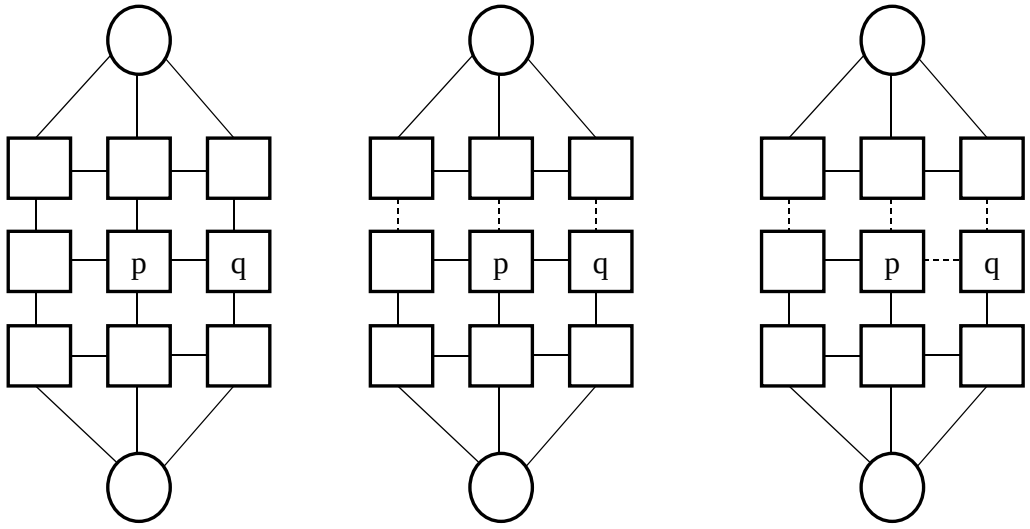
the general form of the energy function we are minimizing.

2.2 Graph cuts

The algorithms presented in this thesis rely on graph cuts as a minimization technique. This section gives basic definitions and notation.

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be a weighted graph where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. $\mathcal{V}$ has two distinguished vertices called the terminals. Usually one of the terminals is called a *source* and the other one is called a *sink*. A *cut* $\mathcal{C} \subset \mathcal{E}$ is a set of edges such that the terminals are separated in the *induced* graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$. In addition, no proper subset of $\mathcal{C}$ separates the terminals in $\mathcal{G}(\mathcal{C})$. Consider for example the graph in figure 2.1(a). Circles denote terminal vertices and squares denote the other vertices. Dashed edges in figure 2.1(b) form a cut because there is no path between terminals if these edges are removed. Dashed edges in figure 2.1(c) do not form a cut because if the dashed edge between vertices p and q is removed the remaining dashed edges still form a cut.

The cost of the cut $\mathcal{C}$, denoted $|\mathcal{C}|$, equals the sum of its edge weights. The minimum cut problem is to find the cut with smallest cost. This problem can



(a) Original graph (b) Dashed edges form a cut (c) Dashed edges don't form a cut

Figure 2.1: Circles show terminal vertices. Squares show the rest of vertices. Dashed lines show edges in the cut.

be solved very efficiently by computing the maximum flow between the terminals, according to a theorem due to Ford and Fulkerson [18]. There are a large number of fast algorithms for this problem (see [1], for example). The worst case complexity is low-order polynomial; however, in practice the running time for the graphs we construct is nearly linear.

2.3 Move spaces

Most of the minimization problems we consider in this work are NP-hard. We approach these problems by finding a local minimum. However in discrete setting, there are many ways to define a local minimum. A natural way to categorize these local minima is through the concept of move spaces, see [32].

We define a local minimum is defined with respect to allowed *moves*. A move is a pair of labelings $(f, f') \in \mathcal{F} \times \mathcal{F}$, where $\mathcal{F}$ is the set of all labelings. Let $\mathcal{M} \subset \mathcal{F} \times \mathcal{F}$ be a set of moves, which we also call a move space. If $(f, f') \in \mathcal{M}$, then we will say that f is within one move from f'. A labeling f is a local minimum with respect to move space $\mathcal{M}$ if $E(f) \leq E(f')$ for any $(f, f') \in \mathcal{M}$. Obviously if $\mathcal{M} = \mathcal{F} \times \mathcal{F}$ then local minimum with respect to $\mathcal{M}$ is also a global minimum.

We construct algorithms which find a local minimum in several interesting move spaces, which are discussed below. For any labeling f there are $O(2^n)$ allowed moves in each of these move spaces. Thus if a labeling is a local minimum with the

3	1	1	1
2	3	1	1
1	2	3	1
1	1	2	3

(a) Labeling f

3	1	1	1
2	3	1	1
2	2	3	1
2	2	2	3

(b) Labeling f'

Figure 2.2: (f, f') is a 1-2 relabel move

respect to these move spaces, it has to satisfy an exponential number of constraints.

Relabel move space

Moves in the relabel move space are conveniently indexed by a pair of labels. Let $\{\alpha, \beta\} \subset \mathcal{L}$. A pair (f, f') is an α-β relabel move if there exists $A \subset \mathcal{P}$ such that

$$\begin{aligned} f_p &= \alpha \quad \text{for } p \in A \\ f'_p &= \beta \quad \text{for } p \in A \\ f_p &= f'_p \quad \text{for } p \notin A. \end{aligned}$$

An α-β relabel move just relabels some set of pixels from α to β. An example of a 1-2 relabel move is shown in figure 2.2. Here A consist of all pixels labeled 1 in the bottom left corner. Finally, the relabel move space is

$$\mathcal{M} = \bigcup_{\{\alpha,\beta\}\subset\mathcal{L}} \{(f,f') \mid (f,f') \text{ is an } \alpha\text{-}\beta \text{ relabel move}\}.$$

Swap move space

Moves in the swap space are called swap moves or simply swaps. Swaps are also indexed by a pair of labels. Let $\{\alpha, \beta\} \subset \mathcal{L}$. A pair (f, f') is a α-β swap if there exists $A \subset \mathcal{P}$ and $B \subset \mathcal{P}$ such that

$$\begin{aligned} f_p = \alpha \text{ and } f'_p = \beta \quad & \text{for } p \in A \\ f_p = \beta \text{ and } f'_p = \alpha \quad & \text{for } p \in B \\ f_p = f'_p \quad & \text{for } p \notin A \bigcup B. \end{aligned}$$

This move is called α-β swap because f' is derived from f by switching labels in A from α to β and labels in B from β to α.

3	1	1	2
2	3	1	1
2	2	3	1
1	2	2	3

(a) Labeling f

3	1	1	1
2	3	1	1
2	2	3	1
2	2	2	3

(b) Labeling f'

Figure 2.3: (f, f') is a 1-2 swap move

3	1	1	2
2	3	1	1
2	2	3	1
2	2	2	3

(a) Labeling f

3	2	2	2
3	3	2	2
3	3	3	2
3	3	3	3

(b) Labeling f'

Figure 2.4: (f, f') is a 1-jump move

An example of a 1-2 swap is shown in figure 2.3. This is a 1-2 swap where A consist of one pixel in the bottom left corner and B consists of one pixel in the upper right corner. Finally, the swap move space is

$$\mathcal{M} = \bigcup_{\{\alpha,\beta\}\subset\mathcal{L}} \{(f,f') \mid (f,f') \text{ is a } \alpha\text{-}\beta \text{ swap}\}\,.$$

Notice that when B is empty then α-β swap is just α-β relabel. Thus relabel move space is contained in swap move space, and it is properly contained in swap move space because, for example, the swap move in figure 2.3 is not a relabel move.

Jump move space

Moves in the jump space are called jump moves or simply jumps. To define the jump move space we need a one to one function $h : \mathcal{L} \to \{0, 1, \ldots, k-1\}$, where k is the number of labels. We define this function because for the jump move space we need the addition operator on labels, and in general labels do not have to be

3	1	1	1
2	3	1	1
2	2	3	1
2	2	2	3

(a) Labeling f

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

(b) Labeling f'

Figure 2.5: (f, f') is an 1-expansion move

numeric quantities. Jumps are labeled by an integer $i \in \{0, 1, \ldots, k-1\}$. A pair (f, f') is an i-jump move if there exists $A \subset \mathcal{P}$ such that

$$\begin{array}{rl} h(f'_p) = h(f_p) + i & \text{for } p \in A \\ h(f'_p) = h(f_p) & \text{for } p \notin A. \end{array} \tag{2.1}$$

A jump moves increases $h(f_p)$ for some pixels p by some fixed amount i. Notice that i can be negative. An example of a jump is shown in figure 2.4. This is a 1-jump move where A consists of all pixels labeled 2 in the lower left corner and all pixels labeled 1. Since labels are natural numbers, h is the identity function in this example. Finally, the jump move space is

$$\mathcal{M} = \bigcup_{i = \{-k+1, \ldots, k-1\}} \{(f,f') \mid (f,f') \text{ is an } i\text{-jump move}\},$$

where k is the number of labels.

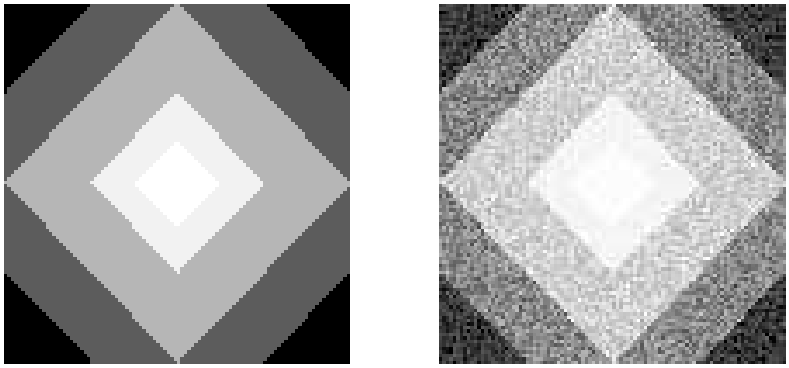
Notice that the relabel move space is properly contained in the jump move space, but the swap and jump move spaces are not contained in each other. For example the swap move in figure 2.3 is not a jump move and the jump move in figure 2.4 is not a swap move.

Expansion move space

Moves in the expansion move space are called expansion moves. Expansion moves are labeled by a single label. Let $\alpha \in \mathcal{L}$. A pair (f, f') is an α-expansion move if there exists $A \subset \mathcal{P}$ and such that

$$\begin{array}{ll} f'_p = \alpha & \text{for } p \in A \\ f_p = f'_p & \text{for } p \notin A \end{array} \tag{2.2}$$

This move is called α-expansion because f' is derived from f by switching all labels in A to α, that is label α is "expanding". An example of an expansion move is



(a) Original image (b) Image corrupted by noise

Figure 2.6: Example of an image restoration problem.

shown in figure 2.5. This is a 1-expansion move where A consist of all pixels in the image. In this example we see that there are fewer moves required to assign a large piece of the image the same label in the expansion move space compared to the swap and jump move spaces. For a labeling in figure 2.3(a) we need to make 2 swaps or 2 jumps to get to the labeling in figure 2.5(b), whereas in the expansion move space we need just one move.

Finally, the expansion move space is

$$\mathcal{M} = \bigcup_{\alpha \in \mathcal{L}} \{(f,f') \mid (f,f') \text{ is } \alpha\text{-expansion move}\}\,.$$

Notice that the expansion move space contains the relabel move space, but the swap, jump, and expansion move spaces are not contained in each other, as shown by the examples in figures 2.3, 2.4, and 2.5.

2.4 Example vision problems

In this section we give examples of two vision problems, image restoration and visual correspondence. We explain what these problems are, show how to state them as labeling problems, and design energy functions to solve them. The performance of the algorithms developed in this thesis will be tested on these problems.

2.4.1 Image restoration

Image restoration is a well studied problem in computer vision. It is simple to formulate and it presents many of the same challenges as other vision problems, such as presence of discontinuities. That is why many energy minimization algo-


(a) Left view of a scene


(b) Right view of a scene

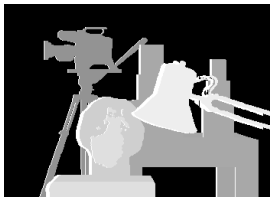

(c) Disparity map

Figure 2.7: Example of a stereo problem.

rithms use it as a test. In addition, real images for this problem are abundant and synthetic images with the desired properties are easy to generate.

In image restoration we are given an image corrupted by noise. The task is to restore the original image. For example figure 2.6(a) shows an image consisting of several regions of constant color. Figure 2.6(b) shows the same image corrupted by white noise. The task is to restore image in figure 2.6(a) from image in figure 2.6(b). This problem arises when the imaging camera is prone to significant noise.

In this case we take $\mathcal{P}$ to be the set of all pixels in the image, and $\mathcal{L}$ is the set of all possible intensities. The energy function is

$$E(f) = \sum_{p} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q).$$

We model $D_p(f_p)$ by

$$D_p(f_p) = (I_p - f_p)^2$$

where I_p is the observed intensity of pixel p. $\mathcal{N}$ is the usual 4-neighbor system, and we will try different types of $V_{\{p,q\}}(f_p, f_q)$ which allow smooth, piecewise constant, and piecewise smooth restoration.

2.4.2 Visual correspondence

Visual correspondence is a fundamental problem in vision. Computation of stereo, motion, optical flow, and other vision problems rely on computing visual correspondence.

In visual correspondence we are given two images of the same real world scene. A pixel in one image is said to correspond to a pixel in another image if the two pixels are projections along lines of sight of the same physical scene element. The problem of visual correspondence is to find pairs of such corresponding pixels. If the images were taken simultaneously from different view points then visual correspondence gives stereo depth of the scene. If images were taken from the same point of view but at different times then visual correspondence gives motion of the scene. The shift in coordinates between the corresponding pixels is called

disparity. In stereo the disparities are usually one dimensional quantities that represent horizontal shifts due to the camera setup or image rectification. In motion, disparities are usually two dimensional and represent shifts in vertical and horizontal directions.

An example of visual correspondence for stereo is given in figure 2.7. Figure 2.7(c) shows the disparity map. The brighter the color, the closer the object is to us. Notice that some pixels in the left image do not correspond to any pixel in the right image. These pixels are shown in the brightest color in (c). Such pixels are said to be *occluded*. In our experiments, we do not allow explicit computation of the occluded pixels, so we assume that they do not exist and compute correspondences for all pixels.

We need to convert visual correspondence into a labeling problem. We choose one of the images to be *primary* and the other one *secondary*. We let $\mathcal{P}$ be the set of pixels in the primary image. The task is to label each pixel in $\mathcal{P}$ with its disparity. Thus the set of labels is the set of all possible disparities.

The energy function is

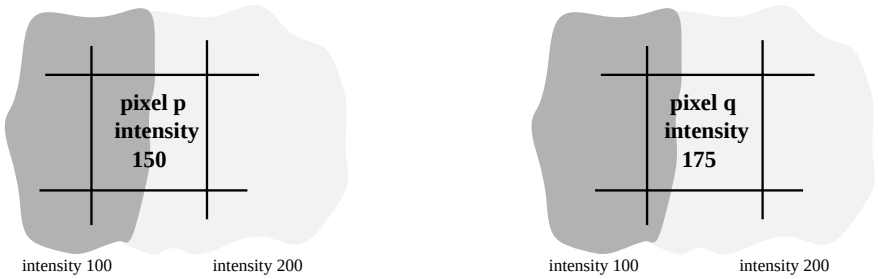
$$E(f) = \sum_{p} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q).$$

$\mathcal{N}$ is again the 4-neighbor system, and we try different types of $V_{\{p,q\}}(f_p, f_q)$ which encode smooth, piecewise constant, and piecewise smooth priors. Our model for $D_p(f_p)$ is more complicated than for image restoration and is discussed in the section below.

Choosing D_p insensitive to image sampling

Let I_p be the intensity of pixel p in the primary image and I'_p be the intensity of pixel p in the secondary image. If pixels p and q correspond, then I_p and I'_q are assumed to be similar. Thus $(I_p - I'_q)^2$ is frequently used as a penalty for deciding that p and q correspond. This penalty has a heavy weight unless $I_p \approx I'_q$. However there are special circumstances when corresponding pixels have very different intensities due to the effects of image sampling. Suppose that the true disparity is not an integer. If a pixel overlaps a scene patch with high intensity gradient, then the corresponding pixels may have significantly different intensities.

When disparities are horizontal shifts, the effect of sampling is shown in figure 2.8. If disparities are integer quantities, pixel p in figure 2.8(a) corresponds to pixel q in figure 2.8(b). Pixel p overlaps the surface patch of intensity 100 and the surface patch of intensity 200 in equal proportions, so its intensity averages to 150. Pixel q overlaps the surface patch of intensity 100 by one quarter and the surface patch of intensity 200 by about three quarters, so its intensity averages to 175. Thus the difference in intensities of p and q is 25, and this is a very large difference to be accounted for by just the camera noise.



(a) Discretization into pixels by the left camera

(b) Discretization into pixels by the right camera

Figure 2.8: The difference in intensities of corresponding pixels p and q is 25 even if there is no camera noise.

For stereo we use the technique in [6] to develop a D_p that is insensitive to image sampling. First we measure how well p fits into the real valued range of disparities $(d - \frac{1}{2}, d + \frac{1}{2})$ by

$$C_{fwd}(p,d) = \min_{d-\frac{1}{2} \le x \le d+\frac{1}{2}} |I_p - I'_{p+x}|$$

where $p + x$ stands for a pixel which has coordinates of p shifted by disparity x. We get fractional values I'_{p+x} by linear interpolation between discrete pixel values. For symmetry we also measure

$$C_{rev}(p,d) = \min_{p-\frac{1}{2} \le x \le p+\frac{1}{2}} |I_x - I'_{p+d}|.$$

$C_{fwd}(p,d)$ can be computed with the following simple formulas which require a few comparisons:

$$L = \min\left\{I'_{p+d}, \frac{I'_{p+d+1} + I'_{p+d}}{2}, \frac{I'_{p+d-1} + I'_{p+d}}{2}\right\}$$

$$U = \max\left\{I'_{p+d}, \frac{I'_{p+d+1} + I'_{p+d}}{2}, \frac{I'_{p+d-1} + I'_{p+d}}{2}\right\}$$

$$C_{fwd}(p,d) = \max\{0, I_p - U, L - I_p\}$$

$C_{rev}(p,d)$ can be computed similarly. The final measure is

$$C(p,d) = \min\{C_{fwd}(p,d), C_{rev}(p,d)\},$$

which is squared to get

$$D_p(d) = C(p,d)^2.$$

For motion we develop a technique similar to [6] except that disparities are now two dimensional. First we measure how well p fits into the two dimensional range of disparities $(d-\frac{1}{2}, d+\frac{1}{2}) \times (d-\frac{1}{2}, d+\frac{1}{2})$ by

$$C_{fwd}(p,d) = \min_{d-\frac{1}{2} \le x \le d+\frac{1}{2}, d-\frac{1}{2} \le y \le d+\frac{1}{2}} |I_p - I'_{p+(x,y)}|$$

Again we get fractional values $I'_{p+(x,y)}$ by linear interpolation between discrete pixel values. For symmetry we also measure

$$C_{rev}(p,d) = \min_{p-\frac{1}{2} \le x \le p+\frac{1}{2}, p-\frac{1}{2} \le y \le p+\frac{1}{2}} |I_x - I'_{p+d}|.$$

$C_{rev}(p,d)$ can be computed with a few comparisons:

$$L = \min \left\{ I_p, \frac{I_p + I_{p+(1,0)}}{2}, \frac{I_p + I_{p+(0,1)}}{2}, \frac{I_p + I_{p+(1,0)} + I_{p+(0,1)} + I_{p+(1,1)}}{4} \right\}$$

$$U = \max \left\{ I_p, \frac{I_p + I_{p+(1,0)}}{2}, \frac{I_p + I_{p+(0,1)}}{2}, \frac{I_p + I_{p+(1,0)} + I_{p+(0,1)} + I_{p+(1,1)}}{4} \right\}$$

$$C_{fwd}(p,d) = \max \left\{ 0, I'_{p+d} - U, L - I'_{p+d} \right\}$$

$C_{fwd}(p,d)$ can be computed similarly. The final measure is

$$C(p,d) = \min \left\{ C_{fwd}(p,d), C_{rev}(p,d) \right\},$$

which is squared to get

$$D_p(d) = C(p,d)^2.$$

Chapter 3

Everywhere smooth prior

In this chapter we design E_{smooth} which encodes a smooth prior and find the exact minimum of the resulting energy by computing a cut on a certain graph. Even though discontinuities are not modeled in E_{smooth}, good results on data with discontinuities are achieved.

3.1 Preliminaries

E_{smooth} is going to express site interactions by a weighted linear function, i.e.

$$V_{(p,q)}(f_p, f_q) = u_{\{p,q\}}|f_p - f_q|.$$

The graph of $V_{(p,q)}(f_p, f_q)$ is shown in figure 1.5. As discussed previously, this function assigns larger penalty for larger differences in labels. The biggest disadvantage of such a function is that the answers are over-smoothed around discontinuities, as we will see on the image restoration example. Another disadvantage is that labels have to be in a correspondence with a subset of integers in some meaningful way. Thus this choice of $V_{(p,q)}(f_p, f_q)$ is inappropriate for motion, for example. In motion labels are two dimensional quantities and cannot be put in correspondence with integers so that linear penalties make sense.

An appropriate choice of coefficients $u_{\{p,q\}}$ which we discuss in section 3.3.1 can help to reduce the problems around discontinuities. Results in section 3.3 show significant improvement when $u_{\{p,q\}}$'s are chosen carefully.

The choice of linear site interaction function yields the following energy to be minimized

$$E_L(f) = \sum_{\{p,q\}\in\mathcal{N}} u_{\{p,q\}}|f_p - f_q| \quad + \quad \sum_{p\in\mathcal{P}} D_p(f_p). \tag{3.1}$$

We will show that the global minimum, i.e. the configuration f that minimizes $E_L(f)$, can be computed by solving a standard two terminal minimum cut problem on a graph.

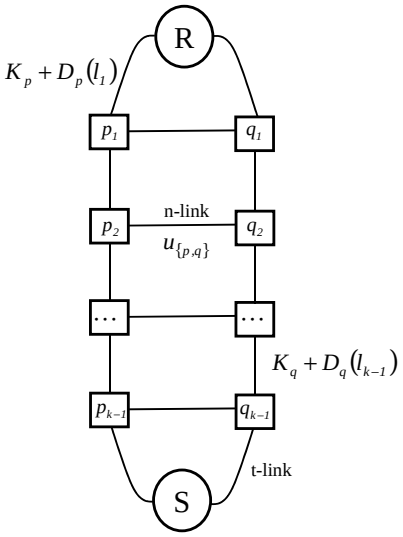


Figure 3.1: A subgraph of $\mathcal{G}$ corresponding to the pixels p and q.

3.2 Minimizing $E_L(f)$

In the proof we will assume without loss of generality that $\mathcal{L}$ consists of consecutive integers. Consider a graph $\mathcal{G}$ defined as follows. There are two terminals: the source R and the sink S. For each pixel p we create a set of vertices $p_1, \ldots, p_{k-1}$ (recall that k is the number of labels). We connect them by edges which we will call t-links $\{t_1^p, \ldots, t_k^p\}$ where $t_1^p = \{R, p_1\}$, $t_j^p = \{p_{j-1}, p_j\}$, and $t_k^p = \{p_{k-1}, S\}$. These t-links are said to correspond to pixel p. For each pair of neighboring pixels p, q and for each $j \in \{1, \ldots, k-1\}$ we create an edge called an n-link $\{p_j, q_j\}$ with weight $u_{\{p,q\}}$. Each t-link t_j^p is assigned a weight $K_p + D_p(l_j)$ where K_p is any constant such that $K_p > (k-1)\sum_{q \in \mathcal{N}_p} u_{\{p,q\}}$. The structure of a subgraph of $\mathcal{G}$ corresponding to a pair of neighboring pixels p and q is shown in figure 3.1.

Definition *A cut is called* feasible *if, for each pixel, it breaks exactly one corresponding t-link.*

Each feasible cut $\mathcal{C}$ corresponds to a configuration which is called $f^{\mathcal{C}}$ in the following way: for each pixel p we take $f_p^{\mathcal{C}} = l_j$ if the t-link t_j^p is cut by $\mathcal{C}$.

Lemma 1 *A minimum cut $\mathcal{C}$ on $\mathcal{G}$ must be feasible.*

PROOF: For each pixel p a cut on $\mathcal{G}$ will break at least one corresponding t-link because the t-links corresponding to p form a path from source to sink. Suppose it breaks more than one, that is suppose t_a^p and t_b^p are cut. Then we can find a smaller cut by restoring t_b^p and breaking n-links $\{p_j, q_j\}$ for all $q \in \mathcal{N}_p$ and $1 \leq j \leq k-1$.

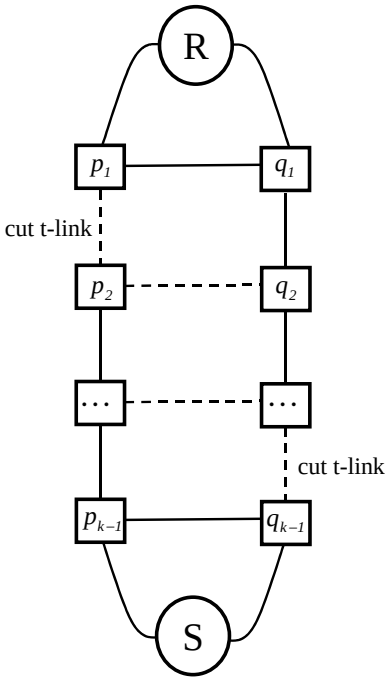


Figure 3.2: All n-links between cut t-links have to be cut.

The cost of the cut will decrease at least by $K_p + D_p(l_b) - (k-1)\sum_{q\in\mathcal{N}_p} w_{\{p,q\}}$ which is strictly positive due to our choice of K_p. ■

Theorem 1 *If $\mathcal{C}$ is a minimum cut on $\mathcal{G}$, then $f^{\mathcal{C}}$ minimizes $E_L(f)$ in (3.1).*

PROOF: It was already shown above that there is a one to one correspondence between the set of feasible cuts and the set of all configurations $f \in \mathcal{F}$. It remains to show that the cost of any feasible cut $\mathcal{C}$ satisfies $|\mathcal{C}| = A + E_L(f^{\mathcal{C}})$, where A is a constant independent of $\mathcal{C}$. If $\mathcal{C}$ is feasible, the cost of cutting t-links is

$$\sum_{p\in\mathcal{P}} \left(K_p + D_p(f_p^{\mathcal{C}})\right). \tag{3.2}$$

A cut does not sever n-links between vertices connected to the same terminal since no proper subset of a cut separates terminals. Therefore for neighboring pixels p and q if $\mathcal{C}$ cuts t_i^p and t_j^q then all the n-links $\{p_h, q_h\}$ for $\min\{i,j\} \le h \le \max\{i,j\}$ are severed, as illustrated in figure 3.2. But there are exactly $|f_p^{\mathcal{C}} - f_q^{\mathcal{C}}|$ such n-links, each weighing $u_{\{p,q\}}$. Thus the total cost of cutting n-links is

$$\sum_{\{p,q\}\in\mathcal{E}_{\mathcal{N}}} u_{\{p,q\}} |f_p^{\mathcal{C}} - f_q^{\mathcal{C}}|. \tag{3.3}$$

Summing up 3.2 and 3.3, we get the cost of $\mathcal{C}$

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}} K_p + \sum_{p \in \mathcal{P}} D_p(f_p^{\mathcal{C}}) + \sum_{\{p,q\} \in \mathcal{E}_{\mathcal{N}}} u_{\{p,q\}} |f_p^{\mathcal{C}} - f_q^{\mathcal{C}}|$$

where $\sum_{p \in \mathcal{P}} K_p$ is a constant independent of $\mathcal{C}$. ■

A graph with a similar structure was first suggested by Cox and Roy [37] for a stereo correspondence problem. The difference between $\mathcal{G}$ and their graph lies in the link weights. Our choice of edge weights guarantees the optimality property of Theorem 1. In contrast, the weights use by Roy and Cox lack theoretical justification. As a result, their algorithm does not appear to have any optimality properties.

Ishikawa and Geiger in [24] describe a stereo algorithm and in [25] describe an image segmentation technique that finds the global minimum of an energy function closely related to $E_L(f)$. Their solution, developed independently before ours, finds a minimum cut on a graph similar to $\mathcal{G}$ except for some details. For example, their graph is directed and has some infinite capacity links, while we employ an undirected graph.

Note that the graph with this structure can be used to encode other everywhere smooth priors. For example if for each neighboring pair of pixels p and q we create n-links $\{p_i, q_j\}$ for all $i, j \in \{1, 2, \ldots, k-1\}$, then the neighbor interaction function is quadratic:

$$V_{\{p,q\}}(f_p, f_q) = (f_p - f_q)^2.$$

However the quadratic neighbor interaction function will over-smooth the answer around discontinuities even more than the linear neighbor interaction function. Out of all neighbor interaction functions which we can encode using this graph construction, the linear neighbor interaction function deals with discontinuities most gracefully. Thus we have chosen not investigate the other possibilities.

3.3 Experimental results

3.3.1 Choosing $u_{\{p,q\}}$ to express contextual information

In this section we discuss how to choose coefficients $u_{\{p,q\}}$ to take advantage of contextual information.

Visual correspondence

The intensities of pixels in the primary image contain information that can significantly influence our assessment of disparities without even considering the secondary image. For example, two neighboring pixels p and q are much more likely to have the same disparity if we know that $I(p) \approx I(q)$, where $I(p)$ and $I(q)$

stand for the intensities of pixels p and q in the primary image. Most methods for computing correspondence do not make use of this kind of contextual information. An exception is Poggio et al. [34], which describes a method based on MRF's. In their approach, intensity edges are used to bias the line process. They allow discontinuities to form without penalty on intensity edges.

We can easily incorporate contextual information into our framework by allowing $u_{\{p,q\}}$ to vary depending on their intensities I_p and I_q. Let

$$u_{\{p,q\}} = U(|I_p - I_q|). \tag{3.4}$$

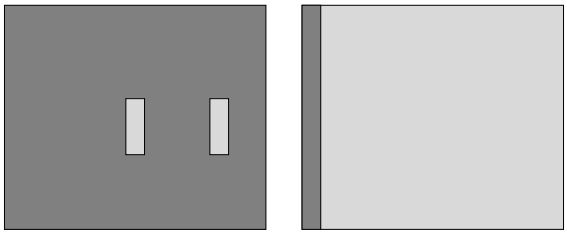
Each $u_{\{p,q\}}$ represents a penalty for assigning different disparities to neighboring pixels p and q. The value of the penalty $u_{\{p,q\}}$ should be smaller for pairs $\{p,q\}$ with larger intensity differences $|I_p - I_q|$. In practice we use an empirically selected decreasing function $U(\cdot)$. Note that instead of (3.4) we could also set the coefficients $u_{\{p,q\}}$ according to an output of an edge detector on the primary image. For example, $u_{\{p,q\}}$ can be made small for pairs $\{p,q\}$ where an intensity edge was detected and large otherwise. Segmentation of the primary image can also be used.

The following example shows the importance of contextual information. Consider the pair of synthetic images below, with a uniformly white rectangle in front of a black background.

Primary image Secondary image

There is a one pixel horizontal shift in the location of the rectangle, and there is no noise. Without noise, the problem of estimating f is reduced to minimizing the smoothness term $E_{smooth}(f)$ under the constraint that pixel p can be assigned disparity d only if $I_p = I'_{p+d}$.

If $u_{\{p,q\}}$ is the same for all pairs of neighbors $\{p,q\}$ then $E_{smooth}(f)$ is minimized at one of the labeling shown in the picture below. Exactly which labeling minimizes $E_{smooth}(f)$ depends on the relationship between the height of the square and the height of the background.



Suppose now that the penalty $u_{\{p,q\}}$ is much smaller if $I_p \neq I_q$ than it is if $I_p = I_q$. In this case the minimum of $E_{smooth}(f)$ is achieved at the disparity configuration shown in the picture below. This result is much closer to human perception.

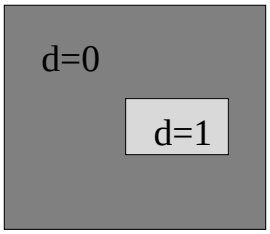


Image restoration

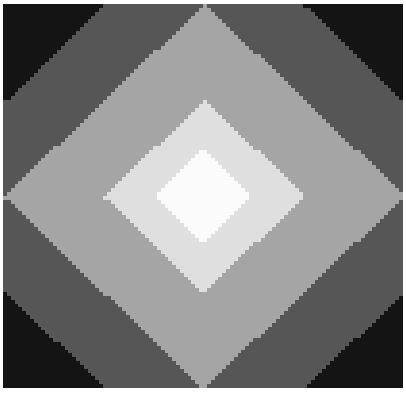
For image restoration one can choose coefficients $u_{\{p,q\}}$ similarly to visual correspondence, although the motivation is somewhat less appealing. In image restoration we observe only one image. Let I_p denote the intensity of the observed image at pixel p. We choose small values of $u_{\{p,q\}}$ for pairs $\{p,q\}$ with very large intensity differences $|I_p - I_q|$. If neighboring pixels have large intensity difference and the noise is not too great, then these two pixels most likely have different intensities in the true image. By choosing small value of $u_{\{p,q\}}$ we will greatly reduce the penalty for assigning p and q labels which are at large distance from each other. In all the experiments we chose $u_{\{p,q\}}$ equal to some small constant if $|I_p - I_q| \leq threshold$ and $u_{\{p,q\}}$ equal to a larger constant otherwise.

3.3.2 Image Restoration examples

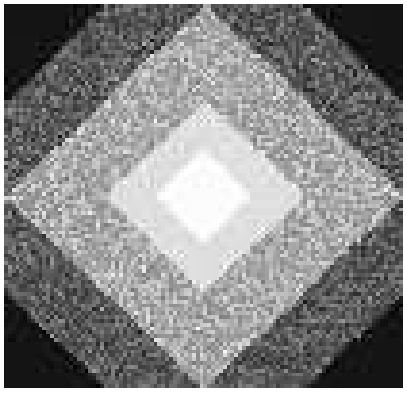
In this section we discuss image restoration results on artificially constructed images. We corrupt images with white Gaussian noise. Since we know the original image, we can quantitively evaluate the results of our algorithm.

We experimented with constant $u_{\{p,q\}}$'s and with variable $u_{\{p,q\}}$'s. For variable $u_{\{p,q\}}$'s we chose

$$u_{\{p,q\}} = \begin{cases} \lambda & \text{if } |I_p - I_q| < 30 \\ 2\lambda & \text{otherwise} \end{cases}$$

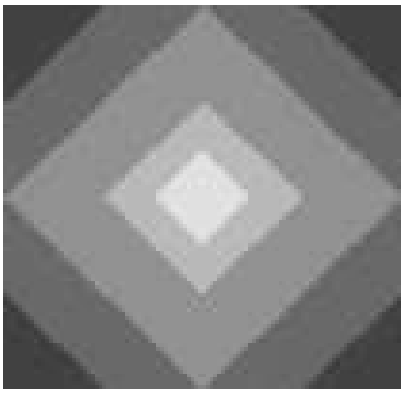


(a) The original image. The intensities of rectangles are 65, 105, 145, 185, and 225.

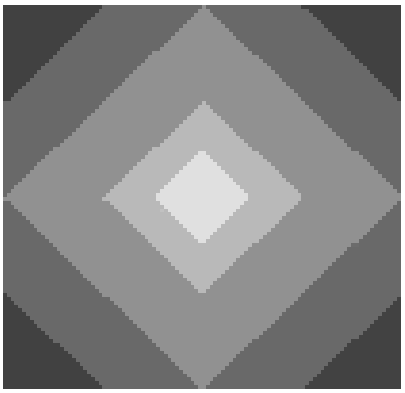


(b) The noisy image. The distribution of noise at each pixel is $N(\mu = 0, \sigma^2 = 16)$.

Figure 3.3: Diamond images.

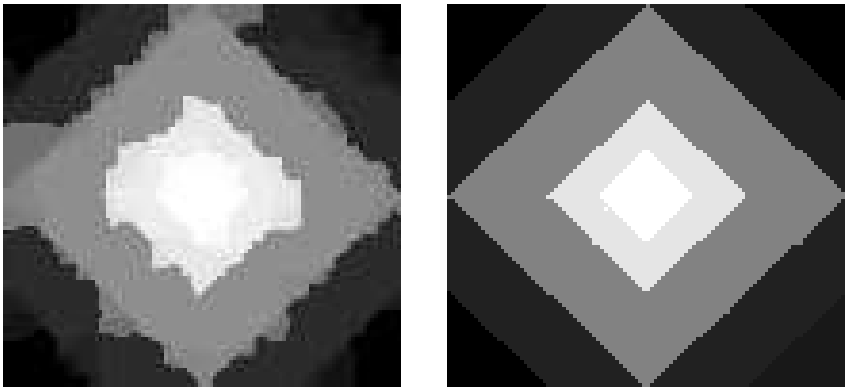


(a) Constant $u_{\{p,q\}}$'s. Average absolute error 0.52.



(b) Variable $u_{\{p,q\}}$'s. Average absolute error 0.06.

Figure 3.4: Results for image restoration example in figure 3.3



(a) Constant $u_{\{p,q\}}$'s (b) Variable $u_{\{p,q\}}$'s

Figure 3.5: Histogram corrected results for image restoration example in figure 3.3

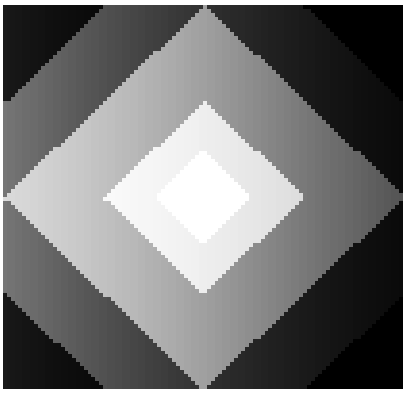
Example 1

Figure 3.3(a) shows the original image consisting of several regions with constant intensities. Figure 3.3(b) shows the same image corrupted by white Gaussian noise. Figure 3.4(a) shows the results of our restoration algorithm when $u_{\{p,q\}} = const$ for the value of *const* which gives the best results (i.e. the chosen *const* minimizes the mean absolute error). We have also experimented with variable $u_{\{p,q\}}$; the results for λ which gives the best results (again the results which minimize the mean absolute error) are shown in figure 3.4(b). Notice that results in figure 3.4(b) are significantly better, as predicted. With variable weights, 96% of pixels were restored correctly and 4% have errors ± 1. With constant weights, 70% of pixels have no error, 20% have error ± 1, the rest have larger errors. To see that there is over-smoothing with constant $u_{\{p,q\}}$'s and no smoothing with variable $u_{\{p,q\}}$'s we histogram corrected[1] both images with the results shown in figure 3.5.

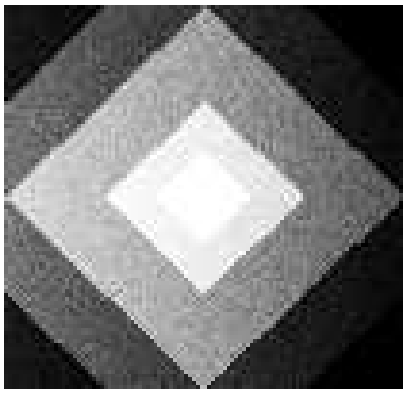
Example 2

Figure 3.6 shows diamond images which consists of several pieces of smoothly varying intensity in each piece. Figure 3.7 shows our results with variable $u_{\{p,q\}}$'s. We see that with variable weights we get pretty sharp intensity edges, and the results within each shaded area vary pretty smoothly. 41% of pixels were found exactly, 46% have ± 1 errors, and the rest of pixels have larger errors.

[1] That is we chose a particular color scheme to make the smoothing effect visible.



(a) The original image



(b) The noisy image. The distribution of noise at each pixel is $N(\mu = 0, \sigma^2 = 16)$.

Figure 3.6: Shaded diamond images.

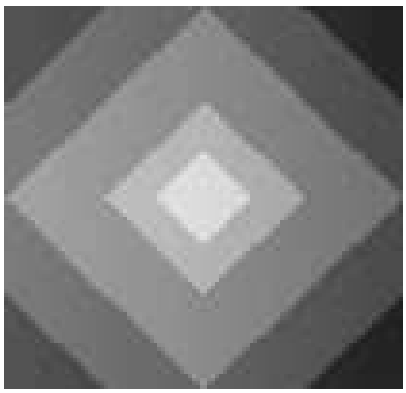
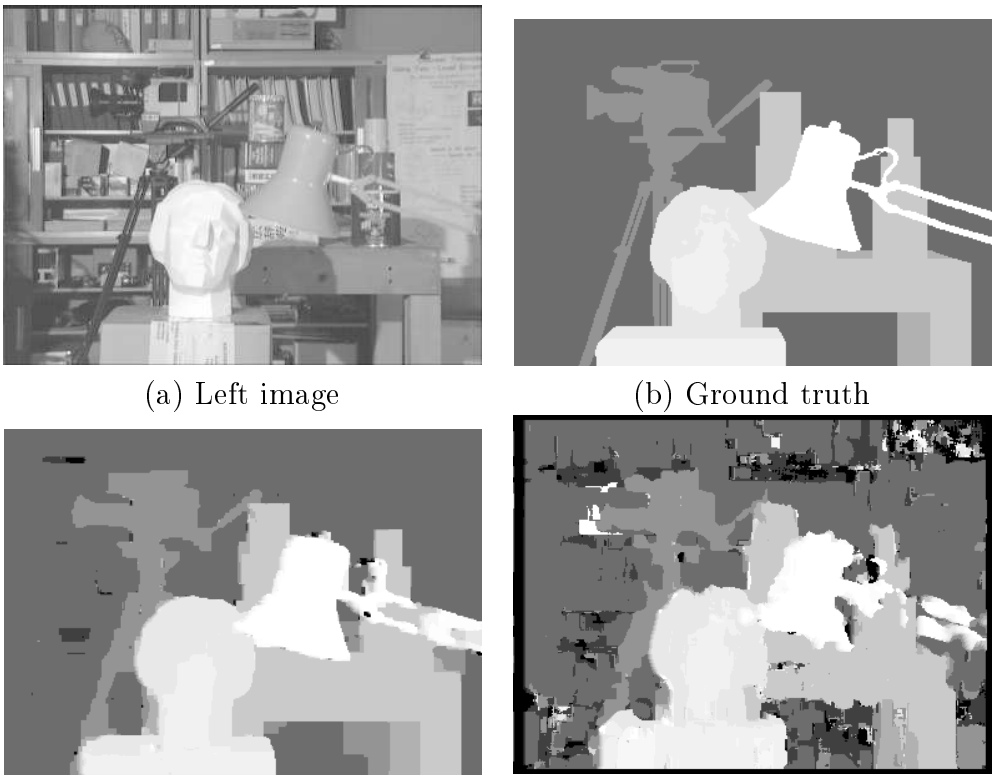


Figure 3.7: Restoration results for image in figure 3.6. Average absolute error is 0.74



(a) Left image (b) Ground truth

(c) Our results (d) Normalized correlation

Figure 3.8: Real imagery with ground truth

3.3.3 Stereo examples

In this section we evaluate our method for several stereo pairs and compare them with the result of normalized correlation.

We chose

$$u_{\{p,q\}} = \begin{cases} \lambda & \text{if } |I_p - I_q| < 5 \\ 2\lambda & \text{otherwise} \end{cases}$$

For normalized correlation we chose parameters which give best statistics when the ground truth is available. If the ground truth is not available we choose parameters which appear to give the best results.

3.3.4 Real imagery with ground truth

Figure 3.8(a) shows the left image of a real stereo pair where the ground truth is known at each pixel. We obtained this image pair from the University of Tsukuba Multiview Image Database. Figure 3.8(b) shows the ground truth, and

Method	% **total errors**	% **errors** $> \pm 1$	**Running time**
Our method	10.1	5.9	8 min
Normalized correlation	24.7	10.0	2 sec

Figure 3.9: Comparison of errors made by our method and normalized correlation

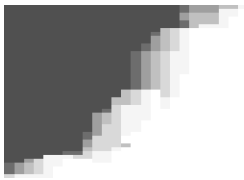


Figure 3.10: Details of disparity map in figure 3.8(a)

figures 3.8(c), (d) show the results of normalized correlation and our algorithm with variable $u_{\{p,q\}}$'s.

Comparison of the errors made by our algorithm and normalized correlation are summarized in figure 3.3.4. Our algorithm does significantly better, we make less than half of total errors and almost half of errors $> \pm 1$ when compared to normalized correlation. However discontinuities are still somewhat over-smoothed. In figure 3.10 we show the enlarged results of our algorithm around the disparity discontinuity between the lamp and the background. Instead of one large drop in disparity between the lamp and background there are several smaller drops in disparity.

The algorithm appears to be stable in the choice of parameters. Figure 3.3.4 summarizes the errors made for different values of parameter λ.

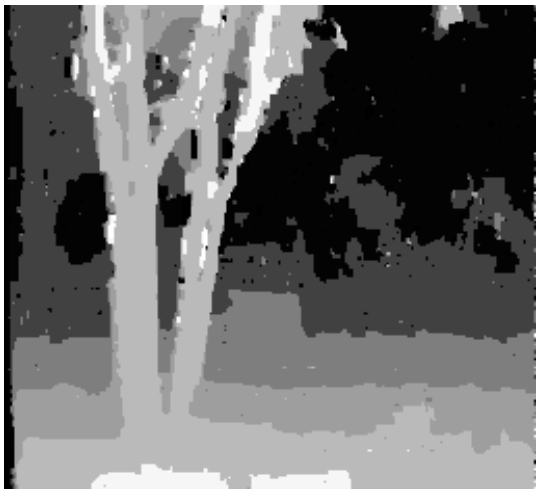
λ	% **of total errors**	% **of errors** $> \pm 1$	**Absolute average error**
5	14.8	5.9	0.22
10	10.1	5.9	0.18
20	10.8	7.5	0.20
50	13.0	9.1	0.23

Figure 3.11: Error statistics for our method for different values of λ

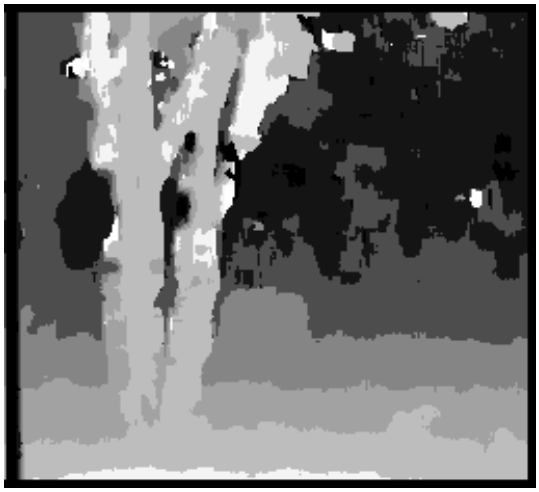
3.3.5 Other real imagery

Figures 3.12 and 3.13 show the results of our algorithm and normalized correlation on standard benchmark image pairs. Our method correctly localized many details in the images. Examples include the tree trunks and the stump in figure 3.12, the front parking meter and the car in figure 3.13. Even fine details such as tree branches in figure 3.12 and the thin pole in figure 3.13 can be discerned in the output.

(a) Left image: size 255 by 233; 8 disparities.



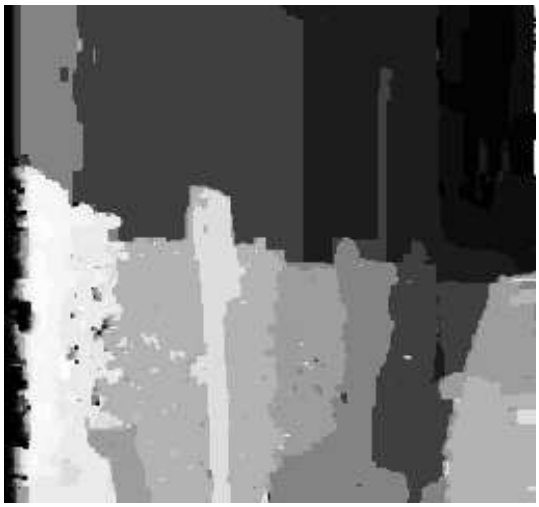
(b) Our answer. Running time 90 sec.



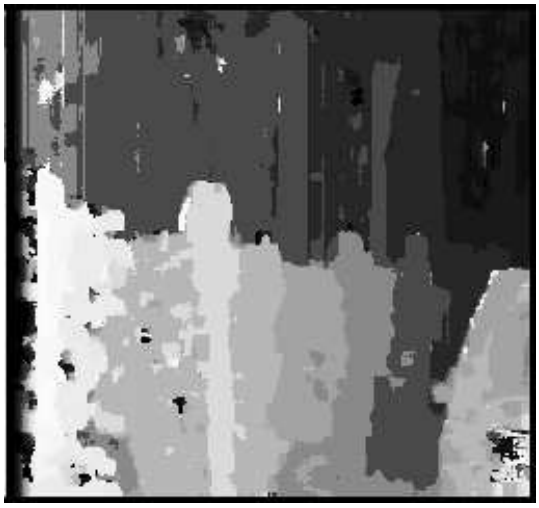
(c) Normalized correlation. Running time 2 sec.

Figure 3.12: SRI tree sequence

(a) Left image: size 256 by 240; 15 disparities



(b) Our answer. Running time 3 min.



(c) Normalized correlation. Running time 3 sec.

Figure 3.13: CMU meter sequence

Chapter 4

Piecewise smooth prior

In this chapter we develop minimization algorithms for energy functions which encode several types of quite general priors. When the neighbor interaction function $V_{\{p,q\}}(f_p, f_q)$ is a semi-metric, we develop an algorithm which finds a local minimum in the swap move space (section 4.3.1). If $V_{\{p,q\}}(f_p, f_q)$ is a metric, we develop an algorithm which find a local minimum in the expansion move space (section 4.3.2). These types of $V_{\{p,q\}}(f_p, f_q)$'s allow us to encode a variety of piecewise smooth priors into the smoothness term. We show in the appendix that minimizing these energy functions is an NP-complete problem in general. We evaluate our methods on the problems of image restoration, stereo, and motion, and get promising results.

4.1 Semi-metric neighbor interactions

Recall that the general form of the energy function we are minimizing is

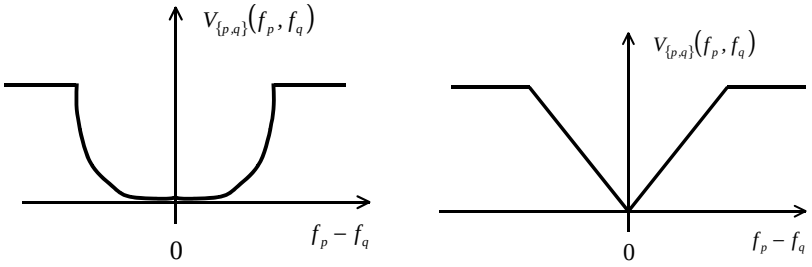
$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q). \tag{4.1}$$

We call $V_{\{p,q\}}(f_p, f_q)$ a *semi-metric* neighbor interaction function if it satisfies the following properties:

$$\begin{array}{lrcl} \text{(i)} & V_{\{p,q\}}(l, l) & = & 0 \\ \text{(ii)} & V_{\{p,q\}}(l_1, l_2) & \geq & 0 \\ \text{(iii)} & V_{\{p,q\}}(l_1, l_2) & = & V_{\{p,q\}}(l_2, l_1). \end{array} \tag{4.2}$$

We are going to denote the energy function in (4.1) with semi-metric $V_{\{p,q\}}$'s by $E_S(f)$. In the appendix we show that minimizing $E_S(f)$ is an NP-complete problem by showing that minimizing even a special case of $E_S(f)$ is an NP-complete problem.

A wide variety of priors can be encoded with a semi-metric neighbor interaction function, but we have only used it for a piecewise smooth prior. Recall



(a) Truncated quadratic (b) Truncated linear

Figure 4.1: Graphs of truncated quadratic and truncated linear $V_{\{p,q\}}$'s.

that to encode a piecewise smooth prior, $V_{\{p,q\}}(f_p, f_q)$ should be nondecreasing in $|f_p - f_q|$, and there should be a limit on how large $V_{\{p,q\}}(f_p, f_q)$ can get. We can encode virtually any piecewise smooth prior with semi-metric $V_{\{p,q\}}$'s, as long as the assumptions in 4.2 are satisfied.

Figure 4.1 shows the graphs of two different $V_{\{p,q\}}$'s we are going to use for the experiments. We call $V_{\{p,q\}}$ in figure 4.1(a) a truncated quadratic:

$$V_{\{p,q\}}(f_p, f_q) = \begin{cases} u_{\{p,q\}}(f_p - f_q)^2 & \text{if } |f_p - f_q| < C \\ u_{\{p,q\}}C^2 & \text{otherwise.} \end{cases}$$

Here C is some constant which sets the bound on the magnitude of $V_{\{p,q\}}$. We call $V_{\{p,q\}}$ in figure 4.1(b) a truncated linear:

$$V_{\{p,q\}}(f_p, f_q) = \begin{cases} u_{\{p,q\}}|f_p - f_q| & \text{if } |f_p - f_q| < C \\ u_{\{p,q\}}C & \text{otherwise.} \end{cases}$$

Again C is a constant limiting the value of $V_{\{p,q\}}$.

4.2 Metric neighbor interactions

In this section we are going to add two additional assumptions about $V_{\{p,q\}}(f_p, f_q)$ to the assumptions in 4.2:

$$\begin{aligned} V_{\{p,q\}}(l_1, l_2) &> 0 \quad \text{if } l_1 \neq l_2, \\ V_{\{p,q\}}(l_1, l_2) + V_{\{p,q\}}(l_2, l_3) &\geq V_{\{p,q\}}(l_1, l_3). \end{aligned} \tag{4.3}$$

Assumptions 4.2 and 4.3 imply that $V_{\{p,q\}}(\cdot, \cdot)$ is a metric. For this reason we are going to call such $V_{\{p,q\}}(f_p, f_q)$ a *metric* neighbor interaction function, and denote the energy function in 4.1 with metric $V_{\{p,q\}}$'s by $E_M(f)$. In the appendix we show that minimizing $E_M(f)$ is an NP-complete problem by showing that minimizing even a special case of $E_M(f)$ is an NP-complete problem.

```
1.  Start with an arbitrary labeling f
2.  Set success := 0
3.  For each i in 𝓘
    3.1.  Find f̂ = arg min E(f') among f' within
          one i-move of f
    3.2.  If E(f̂) < E(f), set f := f̂
          and success := 1
4.  If success = 1 goto 2
5.  Return f
```

Figure 4.2: Algorithm to find a local minimum in move space $\mathcal{M}$.

The range of piecewise smooth priors we can encode with metric $V_{\{p,q\}}$'s is more limited than with semi-metric $V_{\{p,q\}}$'s. For example the truncated quadratic $V_{\{p,q\}}(f_p, f_q)$ in figure 4.1(a) is not a metric:

Let

$$u_{\{x,y\}} = u_{\{y,z\}} = u_{\{x,z\}} = 1$$
$$f_x = 1, \quad f_y = 2, \quad f_z = 3, \quad \text{and } C = 10.$$

Then

$$V_{u_{\{x,y\}}}(f_x, f_y) + V_{u_{\{y,z\}}}(f_y, f_z) = 2 < 4 = V_{u_{\{x,z\}}}(f_x, f_z)$$

However $V_{\{p,q\}}$ in figure 4.1(b) is a metric:

$$V_{u_{\{x,y\}}}(f_x, f_y) + V_{u_{\{y,z\}}}(f_y, f_z) = \min\{|f_x - f_y| + |f_y - f_z|,\ C\} \geq |f_x - f_z|$$

We are going to use the truncated linear $V_{\{p,q\}}$'s for the expansion move space algorithm.

4.3 Local energy minimization

In this section we are going to discuss the general structure of the algorithms to find a local minimum of energy $E_S(f)$ in the swap move space, and of energy $E_M(f)$ in expansion move space. Finding a local minimum in the swap or expansion move spaces is not an easy task, since there is an exponential number of moves possible from each labeling f. Even to check that f is a local minimum is not trivial. Since there is no difference in the general structure of the algorithms for $E_M(f)$ and $E_S(f)$, we are going to refer to these energies by a common name $E(f)$.

As shown in section 2.3, each of the above move spaces can be written as

$$\mathcal{M} = \bigcup_{i \in \mathcal{I}} \{(f,f') \mid (f,f') \text{ is } i\text{-move})\},$$

where $\mathcal{I}$ is some index set whose size depends on the number of labels.

The structure of the algorithm to find a local minimum in $\mathcal{M}$ is outlined in figure 4.2. We start with some arbitrary labeling f. The idea is to keep searching for a labeling f' within one move from f which decreases the value of the energy function, until no such f' can be found. However instead of finding an arbitrary such labeling f', we will find $\hat{f}$ which is within one i-move from f, and which gives the lowest value of the energy function among all labelings f' within one i-move from f. Such a move will be called an optimal i-move from f. When f is obvious, we will just call it the optimal i-move. Finding an optimal i-move from f is the crucial step of the algorithm and it is performed on line 3.1.

Thus the algorithm is greedy: for each fixed i we make the best possible i-move, if it decreases the value of the energy function. The greedy nature of the algorithm will result in its fast convergence. Another advantage of being able to find the optimal i-move efficiently is that it gives a fast check for a local minimum in $\mathcal{M}$: to check if f is a local minimum, it is enough to check that for all i the optimal i-move from f does not decrease the value of the energy function.

We will call a single execution of steps 3.1–3.2 an *iteration*, and an execution of steps 2–4 a *cycle*. In each iteration we find an optimal i-move from the current labeling f. Suppose this move has the form $(f,\hat{f})$. If $E(\hat{f}) < E(f)$ we set $\hat{f}$ to be the current labeling and say that a cycle is successful. Using the index set $\mathcal{I}$, in each cycle we systematically walk over the move space in search for optimal moves which give an improvement. That is in each cycle we perform an iteration for every $i \in \mathcal{I}$ in a certain order that can be fixed or random. The algorithm stops after the first unsuccessful cycle since no further improvement is possible. The final labeling f is a local minimum in the move space $\mathcal{M}$.

4.3.1 Swap move space

In this section we explain how to find an optimal swap move for the energy function $E_S(f)$. Recall that the swap move space is indexed by a pair of labels $\{\alpha, \beta\}$. Thus given a labeling f and a pair of labels $\{\alpha, \beta\}$, the task is to find an optimal α-β swap move from f. The technique is based on computing a cut on a certain graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Its structure will be dynamically determined by a labeling f and a pair of labels $\{\alpha, \beta\}$.

This section is organized as follows. First we describe the construction of $\mathcal{G}$. We show that cuts C on $\mathcal{G}$ correspond in a natural way to labelings $f^{\mathcal{C}}$ which are within one α-β swap move of f. Theorem 1 shows that the cost of a cut is $|\mathcal{C}| = E_S(f^{\mathcal{C}})$ plus a constant. A corollary from this theorem states our main result that the desired labeling $\hat{f}$ equals $f^{\mathcal{C}}$ where $\mathcal{C}$ is a minimum cut on $\mathcal{G}$.

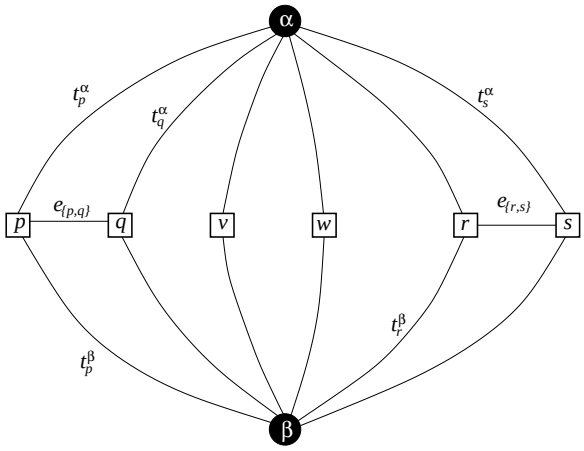


Figure 4.3: An example of the graph $\mathcal{G}$ for a 1D image. Set of the pixels $\mathcal{P} = \{p, q, v, w, r, s\}$. The neighborhood system consists of two pixel pairs $\mathcal{N} = \{\{p, q\}, \{r, s\}\}$, and so there are n-links between pixels p, q and r, s.

The structure of the graph is illustrated in Figure 4.3. For legibility, this figure shows the case of a 1D image. The structure of $\mathcal{G}$ will be as follows. Let $S = \{p \mid f_p \in \{\alpha, \beta\}\}$. The set of vertices includes the two terminals α and β, as well as S. Thus, the set of vertices is

$$\mathcal{V} \;=\; \{\alpha, \beta\} \cup S.$$

Each pixel $p \in S$ is connected to the terminals α and β by edges t_p^α and t_p^β, respectively. For brevity, we will refer to these edges as t-links (terminal links). Each pair of pixels $\{p, q\} \subset S$ which are neighbors (i.e. $\{p, q\} \in \mathcal{N}$) is connected by an edge $e_{\{p,q\}}$ which we will call an n-link (neighbor link). Thus the set of edges consists of n-links and t-links

$$\mathcal{E} = \left(\bigcup_{p \in S} \{t_p^\alpha, t_p^\beta\} \right) \cup \left(\bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ p,q \in S}} e_{\{p,q\}} \right).$$

The weights assigned to the edges are as shown below:

edge	**weight**	**for**
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(\alpha, f_q)$	$p \in S$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(\beta, f_q)$	$p \in S$
$e_{\{p,q\}}$	$V_{\{p,q\}}(\alpha, \beta)$	$\substack{\{p,q\} \in \mathcal{N} \\ p,q \in S}$

Lemma 2 *Any cut $\mathcal{C}$ on $\mathcal{G}$ must sever (include) exactly one t-link for every pixel $p \in S$.*

PROOF: Suppose both $t_p^\alpha \notin \mathcal{C}$ and $t_p^\beta \notin \mathcal{C}$. Then there is a path between the terminals through t-links t_p^α and t_p^β. Now suppose $t_p^\alpha \in \mathcal{C}$ and $t_p^\beta \in \mathcal{C}$. If there is a path from p to terminal α in $\mathcal{G}(C) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$, then $\mathcal{C} - t_p^\alpha$ is still a cut. If there is a path from p to terminal β in $\mathcal{G}(C) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$, then $\mathcal{C} - t_p^\beta$ is still a cut. If there is no path from p to either terminals, then both $\mathcal{C} - t_p^\alpha$ and $\mathcal{C} - t_p^\beta$ are still cuts. ■

Lemma 2 gives an obvious way to define a labeling $f^{\mathcal{C}}$ corresponding to a cut $\mathcal{C}$ on $\mathcal{G}$,

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if } t_p^\alpha \in \mathcal{C} \text{ for } p \in S \\ \beta & \text{if } t_p^\beta \in \mathcal{C} \text{ for } p \in S \\ f_p & \text{for } p \in \mathcal{P}, p \notin S\,. \end{cases} \tag{4.4}$$

In other words, if pixel p is in S then p is assigned label α when the cut $\mathcal{C}$ separates p from the terminal α; similarly, p is assigned label β when $\mathcal{C}$ separates p from the terminal β. If pixel p is not in S then we keep its initial label f_p.

Lemma 3 *A labeling $f^{\mathcal{C}}$ corresponding to a cut $\mathcal{C}$ on $\mathcal{G}$ is one α-β swap away from the initial labeling f.*

PROOF: $(f, f^{\mathcal{C}})$ is a α-β swap since we get $f^{\mathcal{C}}$ from f by switching some pixels in f which are labeled α to β and by switching some pixels in f which are labeled β to α. ■

It is easy to show that a cut $\mathcal{C}$ severs an n-link $e_{\{p,q\}}$ between neighboring pixels on $\mathcal{G}$ if and only if $\mathcal{C}$ leaves the pixels p and q connected to different terminals. Formally

Property 1 *For any cut $\mathcal{C}$ and for any n-link $e_{\{p,q\}}$:*

$$\begin{array}{llll} a) & \textit{If} & t_p^\alpha, t_q^\alpha \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ b) & \textit{If} & t_p^\beta, t_q^\beta \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ c) & \textit{If} & t_p^\beta, t_q^\alpha \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \\ d) & \textit{If} & t_p^\alpha, t_q^\beta \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \end{array}$$

Properties (a) and (b) follow from the requirement that no proper subset of $\mathcal{C}$ should separate the terminals. Properties (c) and (d) also use the fact that a cut has to separate the terminals.

These properties are illustrated in figure 4.4. The following lemma is a consequence of property 1 and equation 4.4.

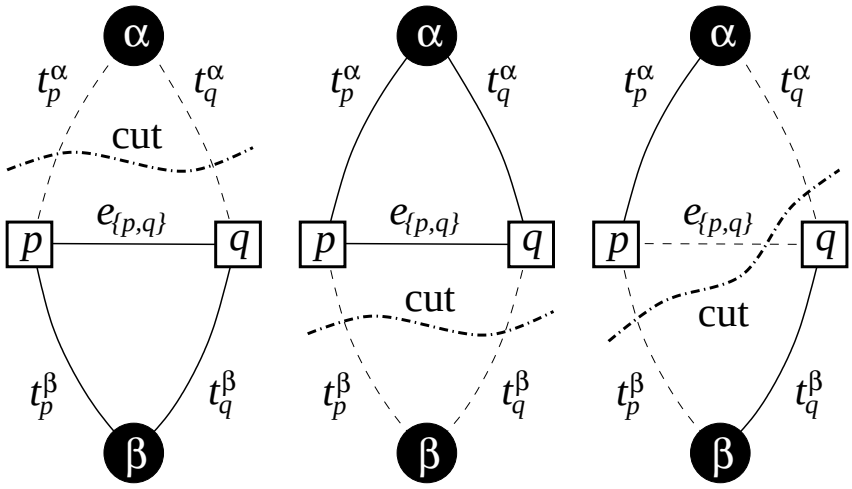


Figure 4.4: Properties of a cut $\mathcal{C}$ on $\mathcal{G}$ for two pixels $p, q \in \mathcal{N}$ connected by an n-link $e_{\{p,q\}}$. Dotted lines show the edges cut by $\mathcal{C}$ and solid lines show the edges remaining in the induced graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$.

Lemma 4 *For any cut $\mathcal{C}$ and for any n-link $e_{\{p,q\}}$*

$$|\mathcal{C} \cap e_{\{p,q\}}| \;=\; V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}).$$

PROOF:
Case 1: $t_p^\alpha, t_q^\alpha \in \mathcal{C}$. Then $e_{\{p,q\}} \notin \mathcal{C}$, and therefore, $|\mathcal{C} \cap e_{\{p,q\}}| = |\emptyset| = 0$. By 4.4 $f_p^{\mathcal{C}} = \alpha$ and $f_q^{\mathcal{C}} = \alpha$, and by 4.2 $V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) = 0 = |\mathcal{C} \cap e_{\{p,q\}}|$.

Case 2: $t_p^\alpha, t_q^\beta \in \mathcal{C}$. In this case, $e_{\{p,q\}} \in \mathcal{C}$ and, therefore, $|\mathcal{C} \cap e_{\{p,q\}}| = |e_{\{p,q\}}| = V_{\{p,q\}}(\alpha, \beta)$. By 4.4, $f_p^{\mathcal{C}} = \alpha$ and $f_q^{\mathcal{C}} = \beta$.
The proofs of the other two cases are similar to the first two cases. ■

Note that this proof assumes that V satisfies all three properties in 4.2, i.e. $V_{\{p,q\}}(\alpha, \alpha) = V_{\{p,q\}}(\beta, \beta) = 0$ and $V_{\{p,q\}}(\alpha, \beta) = V_{\{p,q\}}(\beta, \alpha) \geq 0$.

Lemmas 3 and 4 plus property 1 yield

Theorem 1 *There is a one to one correspondence between cuts $\mathcal{C}$ on $\mathcal{G}$ and labelings that are one α-β swap from f. Moreover, the cost of a cut $\mathcal{C}$ on $\mathcal{G}$ is $|\mathcal{C}| = E_S(f^{\mathcal{C}})$ plus a constant.*

PROOF: We have already shown that a cut $\mathcal{C}$ corresponds to a labeling $f^{\mathcal{C}}$ which is one α-β swap from f in lemma 3. The severed t-links uniquely determine the n-links that must be cut. Therefore for any two distinct cuts $\mathcal{C}_1$ and $\mathcal{C}_2$ there is a

t-link t' such that $t' \in \mathcal{C}_1$ but $t' \notin \mathcal{C}_2$. Therefore $f^{\mathcal{C}_1} \neq f^{\mathcal{C}_2}$, which establishes the desired one to one correspondence.

The cost of a cut $\mathcal{C}$ is

$$|\mathcal{C}| = \sum_{p \in S} |\mathcal{C} \cap \{t_p^\alpha, t_p^\beta\}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \subset S}} |\mathcal{C} \cap e_{\{p,q\}}|. \tag{4.5}$$

Note that for $p \in S$ we have

$$\begin{aligned} |\mathcal{C} \cap \{t_p^\alpha, t_p^\beta\}| &= \begin{cases} |t_p^\alpha| & \text{if } t_p^\alpha \in \mathcal{C} \\ |t_p^\beta| & \text{if } t_p^\beta \in \mathcal{C} \end{cases} \\ &= D_p(f_p^{\mathcal{C}}) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q). \end{aligned}$$

Lemma 4 gives the second term in 4.5. Thus, the total cost of a cut $\mathcal{C}$ is

$$\begin{aligned} |\mathcal{C}| &= \sum_{p \in S} D_p(f_p^{\mathcal{C}}) + \sum_{p \in S} \sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q) \\ &\quad + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \subset S}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) \\ &= \sum_{p \in S} D_p(f_p^{\mathcal{C}}) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ p \text{ or } q \in S}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}). \end{aligned}$$

This can be rewritten as $|\mathcal{C}| = E(f^{\mathcal{C}}) - K$ where

$$K = \sum_{p \notin S} D_p(f_p) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \cap S = \emptyset}} V_{\{p,q\}}(f_p, f_q)$$

is the same constant for all cuts $\mathcal{C}$. ■

Our main result is the corollary from the above theorem:

Corollary 1 *The optimal α-β swap from f is $\hat{f} = f^{\mathcal{C}}$ where $\mathcal{C}$ is the minimum cut on $\mathcal{G}$.*

4.3.2 Expansion move space

In this section we explain how to find an optimal expansion move for energy $E_M(f)$. Recall that the expansion move space is indexed by a single label. Thus given a labeling f and a label α the task is to find an optimal α-expansion move from f. The technique is based on computing a cut on a certain graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Its structure will be dynamically determined by a labeling f and a label α.

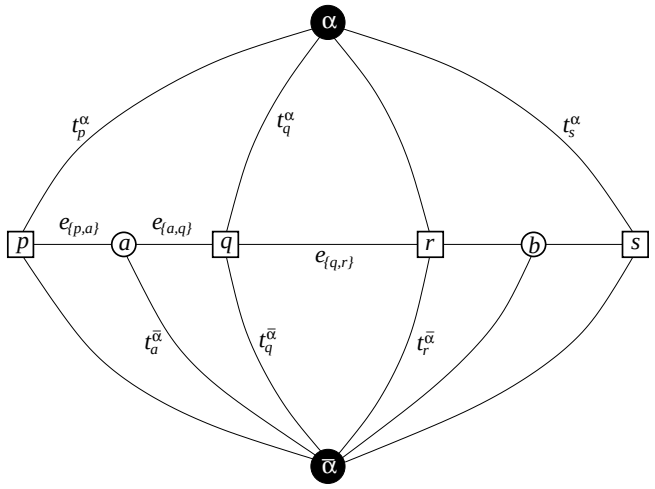


Figure 4.5: An example of the graph $\mathcal{G}$ for a 1D image. The set of pixels in the image is $\mathcal{P} = \{p, q, r, s\}$. The neighborhood system is $\mathcal{N} = \{\{p, q\}, \{q, r\}, \{r, s\}\}$. $f_p \neq f_q$ and $f_r \neq f_s$ therefore there are auxiliary nodes a and b between neighbor pairs $\{p, q\}$ and $\{r, s\}$. $f_q = f_r$ and so there is no auxiliary node between q and r.

This section is organized as follows. First we describe the construction of $\mathcal{G}$ for a given f and α. We show that cuts $\mathcal{C}$ on $\mathcal{G}$ correspond in a natural way to labelings $f^{\mathcal{C}}$ that are within one α-expansion move of f. Then, based on a number of simple properties, we define a class of *elementary* cuts. Theorem 2 shows that elementary cuts are in one to one correspondence with the set of labelings that are within one α-expansion of f, and also that the cost of an elementary cut is $|\mathcal{C}| = E_M(f^{\mathcal{C}})$. A corollary from this theorem states our main result that the desired labeling $\hat{f}$ equals $f^{\mathcal{C}}$ where $\mathcal{C}$ is a minimum cut on $\mathcal{G}$.

The structure of the graph is illustrated in Figure 4.5. For legibility, this figure shows the case of a 1D image. The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all pixels $p \in \mathcal{P}$. In addition, for each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $f_p \neq f_q$ we create an *auxiliary vertex* $a_{\{p,q\}}$. We call these vertices auxiliary because they do not correspond to pixels in $\mathcal{P}$ but provide an auxiliary structure to achieve the desired goal. Thus, the set of vertices is

$$\mathcal{V} \;=\; \{\alpha\} \cup \{\bar{\alpha}\} \cup \mathcal{P} \cup \left(\bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} a_{\{p,q\}} \right).$$

Each pixel $p \in \mathcal{P}$ is connected to the terminals α and $\bar{\alpha}$ by edges t_p^α and $t_p^{\bar{\alpha}}$, correspondingly. We will refer to these edges as t-links (terminal links). Each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $f_p = f_q$ is connected by an edge $e_{\{p,q\}}$ which we will call an n-link (neighborhood link). For each pair of neighboring

pixels $\{p,q\} \in \mathcal{N}$ such that $f_p \neq f_q$ we create a triplet of edges

$$\mathcal{E}_{\{p,q\}} = \left\{ e_{\{p,a\}},\ e_{\{a,q\}},\ t_a^{\bar{\alpha}} \right\},$$

where $a = a_{\{p,q\}}$ is the corresponding auxiliary node. The n-links $e_{\{p,a\}}$ and $e_{\{a,q\}}$ connect pixels p and q to $a_{\{p,q\}}$ and the t-link $t_a^{\bar{\alpha}}$ connects the auxiliary node $a_{\{p,q\}}$ to the terminal $\bar{\alpha}$. Finally, we can write the set of all edges as

$$\mathcal{E} \ = \ \left(\bigcup_{p \in \mathcal{P}} \{t_p^{\alpha}, t_p^{\bar{\alpha}}\} \right) \ \cup \ \left(\bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \right) \ \cup \ \left(\bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right).$$

The weights assigned to the edges are shown in the table below.

edge	**weight**	**for**
$t_p^{\bar{\alpha}}$	∞	p s.t. $f_p = \alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	p s.t. $f_p \neq \alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V_{\{p,q\}}(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$e_{\{a,q\}}$	$V_{\{p,q\}}(\alpha, f_q)$	
$t_a^{\bar{\alpha}}$	$V_{\{p,q\}}(f_p, f_q)$	
$e_{\{p,q\}}$	$V_{\{p,q\}}(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Lemma 2 which states that a cut must sever exactly one t-link for every pixel obviously also holds for this graph construction. This lemma gives a natural labeling $f^{\mathcal{C}}$ corresponding to a cut $\mathcal{C}$ on $\mathcal{G}$. Formally,

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if} \quad t_p^{\alpha} \in \mathcal{C} \\ f_p & \text{if} \quad t_p^{\bar{\alpha}} \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P}. \tag{4.6}$$

In other words, a pixel p is assigned label α if the cut $\mathcal{C}$ disconnects p from the terminal α and, p is assigned its old label f_p if $\mathcal{C}$ disconnects p from the terminal $\bar{\alpha}$. The terminal α stands for the new label and the terminal $\bar{\alpha}$ stands for the old labels assigned to pixels in the initial labeling f.

Lemma 5 *A cut $\mathcal{C}$ on $\mathcal{G}$ corresponds to a labeling $f^{\mathcal{C}}$ which is one α-expansion away from the original labeling f.*

PROOF: A cut $\mathcal{C}$ cannot sever t-links $t_p^{\bar{\alpha}}$ for any pixel p s.t. $f_p = \alpha$ due to the infinite cost. Thus, $f_p^{\mathcal{C}} = \alpha$ for any p which has label α in f. All the other pixels either retain their old label or get assigned the label α in $f_p^{\mathcal{C}}$. ■

If p and q are neighboring pixels (that is $\{p,q\} \in \mathcal{N}$) such that $f_p = f_q$ then we have property 2.

Property 2 *For any cut $\mathcal{C}$ and for any n-link $e_{\{p,q\}}$:*

$$\begin{array}{llll} a) & \textit{If} & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ b) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ c) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\alpha} \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \\ d) & \textit{If} & t_p^{\alpha}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \end{array}$$

The property above is proved exactly like property 1 in section 4.3.1 with β replaced by $\bar{\alpha}$.

Lemma 6 *If $\{p,q\} \in \mathcal{N}$ and $f_p = f_q$ then any cut $\mathcal{C}$ on $\mathcal{G}$ satisfies*

$$|\mathcal{C} \cap e_{\{p,q\}}| = V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}).$$

PROOF: The equation follows from property 2 above, equation 4.6, and the edge weights. ■

Consider now the triplet set of edges $\mathcal{E}_{\{p,q\}}$ corresponding to a pair of neighboring pixels $\{p,q\} \in \mathcal{N}$ such that $f_p \neq f_q$. In this case, there are several different ways to cut these edges even when the pair of severed t-links at p and q is fixed. However, a minimum cut $\mathcal{C}$ on $\mathcal{G}$ is guaranteed to sever the edges in $\mathcal{E}_{\{p,q\}}$ in a unique way depending on what t-links are cut at the pixels p and q.

The rule for this case is described in property 3 below. Assume that $a = a_{\{p,q\}}$ is an auxiliary node between the corresponding pair of neighboring pixels.

Property 3 *A minimum cut cut $\mathcal{C}$ on $\mathcal{G}$ satisfies:*

$$\begin{array}{llll} a) & \textit{If} & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ b) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a^{\bar{\alpha}}. \\ c) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\alpha} \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \\ d) & \textit{If} & t_p^{\alpha}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \end{array}$$

Property(a) results from the fact that no subset of $\mathcal{C}$ is a cut. The others follow from the minimality of $|\mathcal{C}|$ and the fact that $e_{\{p,a\}}$, $e_{\{a,q\}}$ and $t_a^{\bar{\alpha}}$ satisfy the triangle inequality so that cutting any one of them is cheaper than cutting the other two together. These properties are illustrated in figure 4.6.

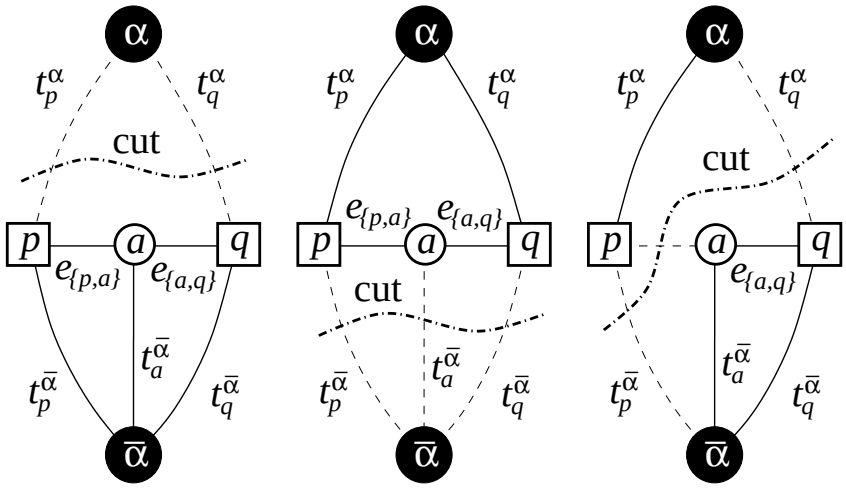


Figure 4.6: Properties of a minimum cut $\mathcal{C}$ on $\mathcal{G}$ for two pixel $p, q \in \mathcal{N}$ such that $f_p \neq f_q$. Dotted lines show the edges cut by $\mathcal{C}$ and solid lines show the edges in the induced graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$.

Lemma 7 *If $\{p, q\} \in \mathcal{N}$ and $f_p \neq f_q$ then the minimum cut $\mathcal{C}$ on $\mathcal{G}_\alpha$ satisfies*

$$|\mathcal{C} \cap \mathcal{E}_{\{p,q\}}| = V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}})$$

PROOF: The equation follows from the property 3 above, equation 4.6, and the edge weights. For example, if $t_p^{\bar\alpha}, t_q^{\bar\alpha} \in \mathcal{C}$ then $|\mathcal{C} \cap \mathcal{E}_{\{p,q\}}| = |t_a^{\bar\alpha}| = V_{\{p,q\}}(f_p, f_q)$. At the same time, 4.6 implies that $f_p^{\mathcal{C}} = f_p$ and $f_q^{\mathcal{C}} = f_q$. ■

Note that the correct penalty $V_{\{p,q\}}$ is imposed whenever $f_p^{\mathcal{C}} \neq f_q^{\mathcal{C}}$. This is exactly what the auxiliary pixel construction was designed for. We had to develop a special trick for the case when the original labels for p and q do not agree ($f_p \neq f_q$) in order to get the same effect that lemma 6 establishes for the simpler situation when $f_p = f_q$.

Property 2 holds for any cut, and property 3 holds for a minimum cut. However, there can be other cuts besides the minimum cut that satisfy these two properties. See for example the cut in figure 4.7. We will define an *elementary* cut on $\mathcal{G}$ to be a cut that satisfies properties 2 and 3.

Theorem 2 *Then there is a one to one correspondence between the set of all elementary cuts on $\mathcal{G}$ and the set of all labelings within one α-expansion of f. Moreover, for any elementary cut $\mathcal{C}$ we have $|\mathcal{C}| = E_M(f^{\mathcal{C}})$.*

PROOF: We first show that an elementary cut $\mathcal{C}$ is uniquely determined by the corresponding labeling $f^{\mathcal{C}}$. The label $f_p^{\mathcal{C}}$ at the pixel p determines which of the t-links to p is in $\mathcal{C}$. Property 2 shows which n-links $e_{\{p,q\}}$ between pairs of neighboring

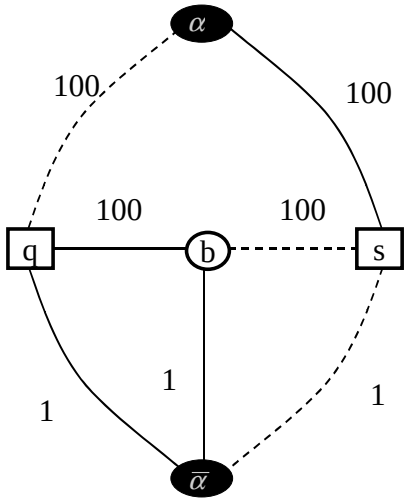


Figure 4.7: The cut is shown in dashed lines. It satisfies properties 2 and 3 but is not the minimum cut on this graph. The minimum cut is $\mathcal{C} = \{t_q^{\bar{\alpha}}, t_s^{\bar{\alpha}}, t_b\}$.

pixels $\{p, q\}$ such that $f_p = f_q$ should be severed. Similarly, property 3 determines which of the links in $\mathcal{E}_{\{p,q\}}$ corresponding to $\{p, q\} \in \mathcal{N}$ such that $f_p \neq f_q$ should be cut.

We now compute the cost of a elementary cut $\mathcal{C}$, which is

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}} |\mathcal{C} \cap \{t_p^{\alpha}, t_p^{\bar{\alpha}}\}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} |\mathcal{C} \cap e_{\{p,q\}}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} |\mathcal{C} \cap \mathcal{E}_{\{p,q\}}|. \quad (4.7)$$

The cost of the cut t-links is

$$|\mathcal{C} \cap \{t_p^{\alpha}, t_p^{\bar{\alpha}}\}| = \begin{cases} |t_p^{\alpha}| & \text{if } t_p^{\alpha} \in \mathcal{C} \\ |t_p^{\bar{\alpha}}| & \text{if } t_p^{\bar{\alpha}} \in \mathcal{C} \end{cases} = \begin{cases} D_p(\alpha) & \text{if } f_p^{\mathcal{C}} = \alpha \\ D_p(f_p) & \text{if } f_p^{\mathcal{C}} = f_p \end{cases} = D_p(f_p^{\mathcal{C}}).$$

Therefore, the first term in 4.7 is

$$\sum_{p \in \mathcal{P}} |\mathcal{C} \cap \{t_p^{\alpha}, t_p^{\bar{\alpha}}\}| = \sum_{p \in \mathcal{P}} D_p(f_p^{\mathcal{C}}).$$

Lemmas 6 and 7 hold for elementary cuts, since they were based on properties 2 and 3. Then lemmas 6 and 7 give us the second and the third terms in 4.7. Thus, the total cost of a elementary cut $\mathcal{C}$ is

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}} D_p(f_p^{\mathcal{C}}) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) = E_M(f^{\mathcal{C}}).$$

■

Our main result is a simple consequence of this theorem, since the minimum cut is an elementary cut.

Corollary 2 *Let $\mathcal{C}$ be the minimum cut on $\mathcal{G}$. Then the optimal labeling $\hat{f} = \arg\min E_M(f')$ among f' within one α-expansion of f is given by $\hat{f} = f^{\mathcal{C}}$.*

4.3.3 Running time

We now discuss the running time of the swap and expansion move space algorithms. The swap and expansion algorithms perform k^2 and k iterations in a cycle, respectively. On each iteration we find a minimum cut on a graph of size $O(n)$, and due to the special structure of this graph, the time to find a cut is effectively linear.

Without making any assumptions, we do not have a good bound on the number of cycles it takes the algorithm to converge. Now assume that $D_p(f_p)$ and $V_{\{p,q\}}(f_p, f_q)$ are independent of the image size. Initialize an algorithm with the following labeling f^s:

$$f_p^s = \arg\min_{l \in \mathcal{L}} D_p(l) \tag{4.8}$$

Without loss of generality we can assume $f_p^s = 0$ because minimizing

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\}} V_{\{p,q\}}(f_p, f_q)$$

is equivalent to minimizing

$$E(f) = \sum_{p \in \mathcal{P}} D'_p(f_p) + \sum_{\{p,q\}} V_{\{p,q\}}(f_p, f_q),$$

where

$$D'_p(f_p) = D_p(f) - \arg\min_{l \in \mathcal{L}} D_p(l).$$

Let

$$c = \max_{\{p,q\} \subset \mathcal{P},\ \{l_1,l_2\} \subset \mathcal{L}} V_{\{p,q\}}(l_1, l_2).$$

This is a constant independent of image size. Thus the algorithm starts with the energy $E(f^s) \leq \sum_{\{p,q\} \in \mathcal{N}} c$.

Let u be the smallest possible difference between coefficients D_p and $V_{\{p,q\}}$. That is

$$u = \min_{a,b \in S} |a - b|,$$

where

$$S = \left\{ V_{\{p,q\}}(l, l') \mid l, l' \in \mathcal{L},\ p \in \mathcal{P} \right\} \cup \left\{ D_p(l) \mid l \in \mathcal{L},\ p \in \mathcal{P} \right\}.$$

It is obvious that u is a constant independent of n.

At each cycle except the last one, the energy must decrease by at least u. Therefore the algorithm must converge in $\sum_{\{p,q\} \in \mathcal{N}} \frac{c}{u}$ cycles, which is $O(n)$. This is a rather large bound, however in all the experiments we performed the algorithms converged in 2-8 cycles (see section 4.4).

4.3.4 Optimality properties

In this section we prove that a local minimum in the expansion move space is within some factor depending on $V_{\{p,q\}}(f_p, f_q)$ from the optimal solution. For notational convenience we will write E instead of E_M.

Let f^e be a local minimum in the expansion move space, and let f^* be the optimal solution. Fix some $\alpha \in \mathcal{L}$ and let

$$\mathcal{P}_\alpha = \left\{ p \in \mathcal{P} \mid f_p^* = \alpha \right\}$$

We can produce a labeling f' within one α-expansion move from f^e as follows:

$$f'_p = \begin{cases} \alpha & \text{if } p \in \mathcal{P}_\alpha \\ f_p^e & \text{otherwise} \end{cases}$$

The key observation is that since f^e is a local minimum in the expansion move space,

$$E(f^e) \leq E(f'). \tag{4.9}$$

Let S be a set of pixels and pairs of neighboring pixels. If p, q denote pixels, define $E_S(f)$ as a restriction of the energy to the set S.
Formally,

$$E_S(f) = \sum_{p \in S} D_p(f_p) + \sum_{\{p,q\} \in S} V_{\{p,q\}}(f_p, f_q).$$

Let I^α be the set of pixels and pairs of neighboring pixels contained inside $\mathcal{P}_\alpha$, B^α be the set of pairs of neighboring pixels on the boundary of $\mathcal{P}_\alpha$, and O^α be the set of pixels and pairs of neighboring pixels contained outside of $\mathcal{P}_\alpha$.
Formally,

$$I^\alpha = \{p \mid p \in \mathcal{P}_\alpha\} \cup \{\{p,q\} \mid \{p,q\} \in \mathcal{N} \text{ and } \{p,q\} \subset \mathcal{P}_\alpha\},$$

$$B^\alpha = \{\{p,q\} \mid \{p,q\} \in \mathcal{N} \text{ and } \{p,q\} \cap \mathcal{P}_\alpha \neq \emptyset \text{ and } \{p,q\} \cap (\mathcal{P} - \mathcal{P}_\alpha) \neq \emptyset\},$$

$$O^\alpha = \{p \mid p \notin \mathcal{P}_\alpha\} \cup \{\{p,q\} \mid \{p,q\} \in \mathcal{N} \text{ and } \{p,q\} \subset (\mathcal{P} - \mathcal{P}_\alpha)\}.$$

Obviously,

$$I^\alpha \cup B^\alpha \cup O^\alpha = \{p \mid p \in \mathcal{P}\} \cup \{\{p,q\} \mid \{p,q\} \in \mathcal{N}\},$$

and so we can break $E(f^e)$, $E(f^\alpha)$, and $E(f^*)$ into energies restricted to sets I^α, B^α, and O^α:

$$E(f^e) = E_{I^\alpha}(f^e) + E_{B^\alpha}(f^e) + E_{O^\alpha}(f^e), \tag{4.10}$$

$$E(f') = E_{I^\alpha}(f') + E_{B^\alpha}(f') + E_{O^\alpha}(f'), \tag{4.11}$$

$$E(f^*) = E_{I^\alpha}(f^*) + E_{B^\alpha}(f^*) + E_{O^\alpha}(f^*). \tag{4.12}$$

Let

$$c = \max_{\{p,q\}\in\mathcal{N}} \left(\frac{\max_{l_1 \neq l_2 \in \mathcal{L}} V_{\{p,q\}}(l_1, l_2)}{\min_{l_3 \neq l_4 \in \mathcal{L}} V_{\{p,q\}}(l_3, l_4)} \right). \tag{4.13}$$

The following three facts hold:

$$E_{O^\alpha}(f') = E_{O^\alpha}(f^e), \tag{4.14}$$

$$E_{I^\alpha}(f') = E_{I^\alpha}(f^*), \tag{4.15}$$

$$E_{B^\alpha}(f') \leq cE_{B^\alpha}(f^*). \tag{4.16}$$

Equations 4.14 and 4.15 are obvious, and equation 4.16 holds because

$$\begin{aligned} cE_{B^\alpha}(f^*) &= \sum_{\{p,q\}\in B^\alpha} cV_{\{p,q\}}(f_p^*, f_q^*) \geq \sum_{\{p,q\}\in B^\alpha} c \min_{l_1,l_2\in\mathcal{L}} V_{\{p,q\}}(l_1, l_2) \geq \\ &\geq \sum_{\{p,q\}\in B^\alpha} \frac{\max_{l_1,l_2\in\mathcal{L}} V_{\{p,q\}}(l_1, l_2)}{\min_{l_1,l_2\in\mathcal{L}} V_{\{p,q\}}(l_1, l_2)} \min_{l_1,l_2\in\mathcal{L}} V_{\{p,q\}}(l_1, l_2) \geq \\ &\geq \sum_{\{p,q\}\in B^\alpha} \max_{l_1,l_2\in\mathcal{L}} V_{\{p,q\}}(l_1, l_2) \geq \max_{f\in\mathcal{F}} E_{B^\alpha}(f) \geq E_{B^\alpha}(f'). \end{aligned}$$

Substituting 4.10 and 4.11 into 4.9 we get:

$$E_{I^\alpha}(f^e) + E_{B^\alpha}(f^e) + E_{O^\alpha}(f^e) \leq E_{I^\alpha}(f') + E_{B^\alpha}(f') + E_{O^\alpha}(f')$$

Using fact 4.14, the above equation simplifies to:

$$E_{I^\alpha}(f^e) + E_{B^\alpha}(f^e) \leq E_{I^\alpha}(f') + E_{B^\alpha}(f').$$

Finally we use facts 4.15 and 4.16 to get a bound on the part of the energy restricted to $I^\alpha \cup O^\alpha$

$$E_{I^\alpha}(f^e) + E_{B^\alpha}(f^e) \leq E_{I^\alpha}(f^*) + cE_{B^\alpha}(f^*). \tag{4.17}$$

To get the bound on the total energy, we need to sum equation 4.17 over all labels $\alpha \in \mathcal{L}$:

$$\sum_{\alpha\in\mathcal{L}} (E_{I^\alpha}(f^e) + E_{B^\alpha}(f^e)) \leq \sum_{\alpha\in\mathcal{L}} (E_{I^\alpha}(f^*) + cE_{B^\alpha}(f^*)) \tag{4.18}$$

Let $B = \bigcup_{\alpha\in\mathcal{L}} B^\alpha$. Observe that for every $\{p, q\} \in B$, $E_{\{p,q\}}(f^e)$ appears twice on the left side of 4.17, once in $E_{B^\alpha}(f^e)$ for $\alpha = f_p^*$ and once in $E_{B^\alpha}(f^e)$ for $\alpha = f_q^*$. Similarly every $E_{\{p,q\}}(f^*)$ appears 2c times on the right side of 4.17. Therefore equation 4.18 can be rewritten to get the bound of $2c$:

$$E(f^e) + E_B(f^e) \leq E(f^*) + (2c - 1)E_B(f^*) \leq 2cE(f^*),$$

and so

$$E(f^e) \leq 2cE(f^*).$$

Note that Kleinberg and Tardos [29] develop an algorithm for minimizing E_M which also has optimality properties. In case of the Potts $V_{\{p,q\}}$ discussed in the next chapter their algorithm has a bound of 2, the same bound as we have. In case of a general metric $V_{\{p,q\}}$ they have a bound of $O(\log k \log\log k)$. Their algorithm uses linear programming, which is impractical for the large number of variables occurring in computer vision.

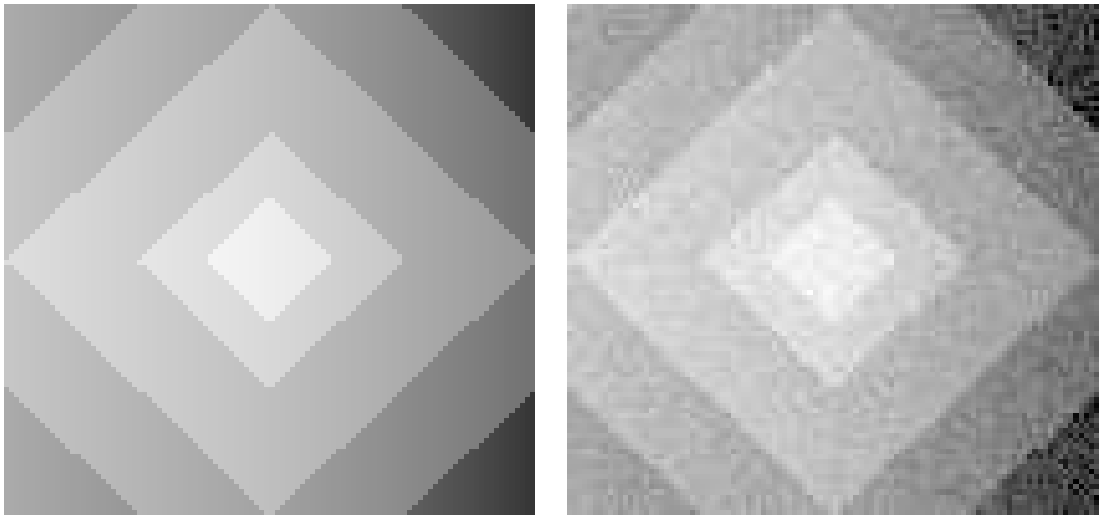
4.4 Experimental results

In this section we present experimental results for image restoration, stereo, and motion. To assess how well our algorithms perform energy minimization, we compare the energy achieved by our algorithms to the energy produced by simulated annealing. To assess how well our algorithms solve the visual correspondence problem, we compare the results to normalized correlation, the standard visual correspondence method.

$D_p(f_p)$'s are chosen as explained in 2.4.2. We experiment with truncated linear and truncated quadratic $V_{\{p,q\}}$'s for the swap algorithm and we use truncated linear $V_{\{p,q\}}$'s for the expansion algorithm. For normalized correlation we chose the parameters which appear to give the best results. For simulated annealing we compared a number of annealing methods and chose the one that worked the best, that is the one that appears to give faster convergence. It turned out to be the "Metropolis-Heatbath" version with truncated logarithmic cooling schedules. To give it a good starting point, simulated annealing was initialized with the input image for the image restoration problem, and with results of normalized correlation for the stereo problem.

4.4.1 Image restoration

Figure 4.8(a) shows an image consisting of large constant-intensity regions which are gradually shaded, as if there were a light source to the left of the image. This image corrupted by $N(0, 100)$ noise is shown in figure 4.8(b). The energy computed by our swap algorithm with truncated quadratic $V_{\{p,q\}}$'s is compared with the energy computed by simulated annealing as a function of time in figure 4.9. Notice that the horizontal axis representing time is on the logarithmic scale. Our algorithm produces very low energy after the first iteration, while annealing decreases the energy very slowly. The energies obtained are summarized in figure 4.10. Although simulated annealing eventually achieves a slightly better energy, it takes an extremely long time to do so. Figure 4.11 shows the final results of our algorithm and simulated annealing.



(a)Original image (b) Image corrupted by $N(0, 100)$

Figure 4.8: An instance of image restoration problem.

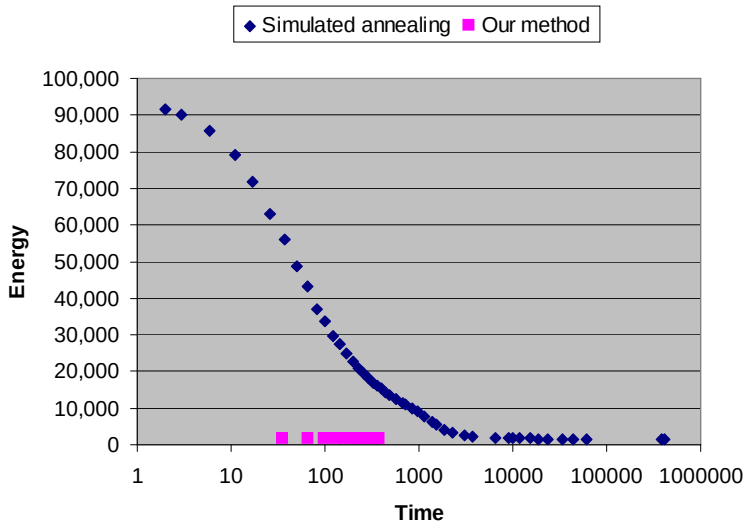
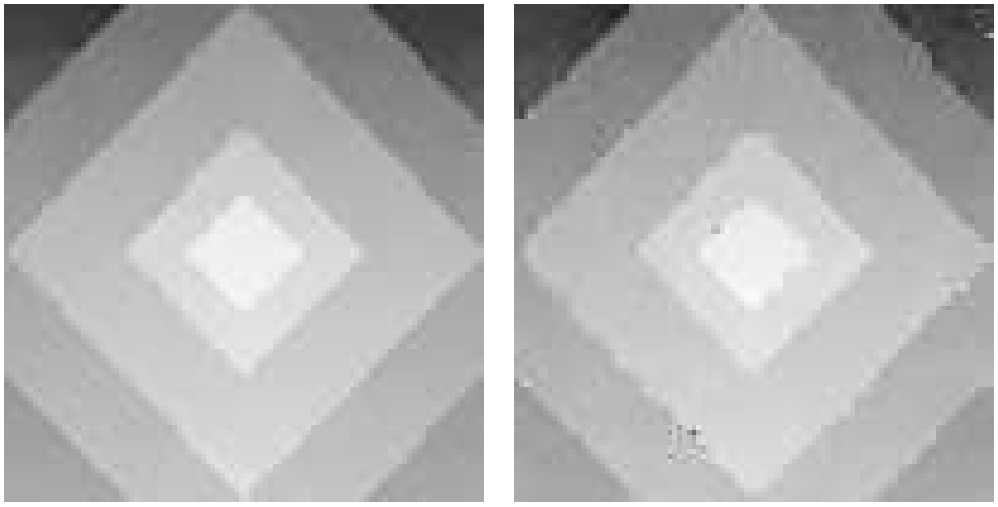


Figure 4.9: Performance comparison of swap method and simulated annealing for the problem of stereo.

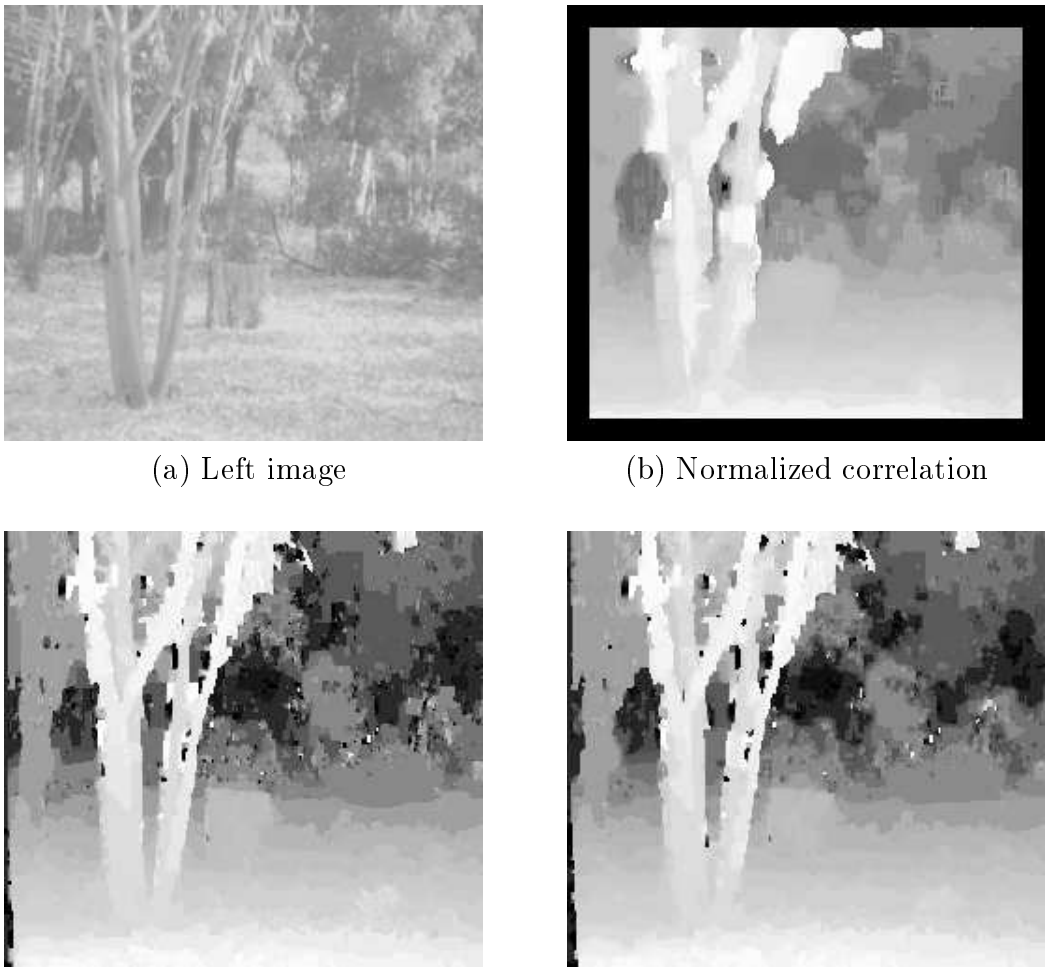
	E		E_{smooth}	
	our results	**annealing**	**our results**	**annealing**
First cycle, $t = 36$	1577	55892	637	9658
Last cycle, $t = 389$	1472	15215	576	8475
Annealing, $t = 417317$	—	1458	—	571

Figure 4.10: Summary of total energy and smoothness energy produced by our algorithm and simulated annealing.



(a)Our method (b) Simulated annealing

Figure 4.11: An instance of image restoration problem.



(a) Left image (b) Normalized correlation

(c) Expansion algorithm (d) Swap algorithm

Figure 4.12: Tree image

4.4.2 Stereo

In this section we compare the performance of the swap and expansion algorithms when $V_{\{p,q\}}$'s are modeled by a truncated linear function.

Figure 4.12(a) shows the left image of a stereo pair, figures 4.12(b),(c) and (d) show the results of normalized correlation, expansion and swap algorithms respectively. Notice that there is a large number of disparities for this stereo pair. The ground consists of a large piece of smoothly varying disparity, but the tree on the foreground form sharp discontinuities with the background. Our methods do quite well in preserving sharp discontinuities while recovering smoothly varying background disparity surface. Figure 4.13 compares the performance of expansion and swap algorithms against each other. The swap algorithm converges faster, but

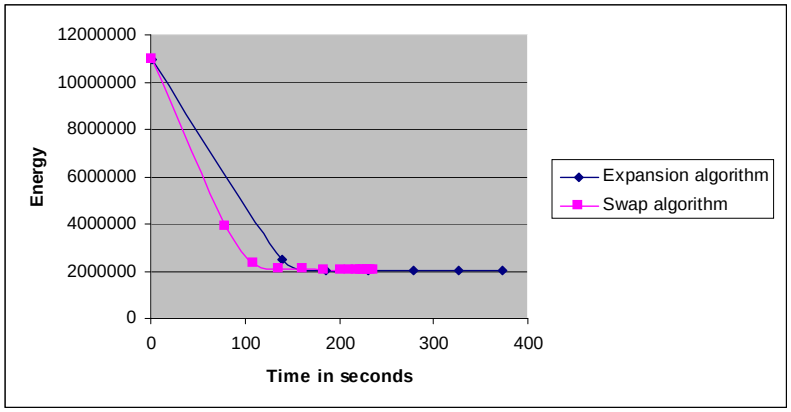
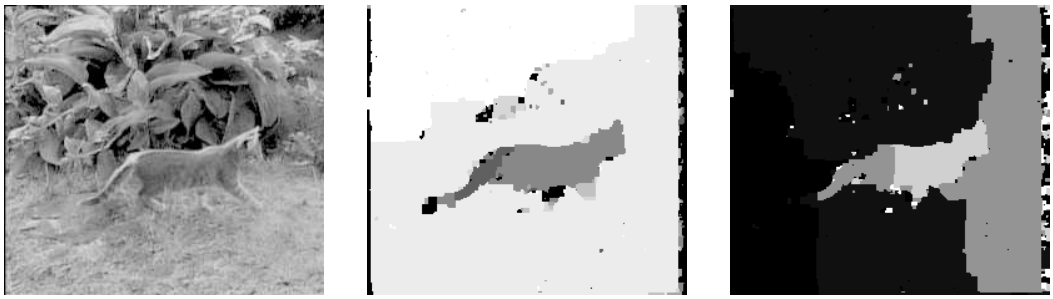


Figure 4.13: Comparison of swap and expansion algorithms.



(a) First image (b) Horizontal movement (c) Vertical movement

Figure 4.14: Moving cat

the expansion algorithm achieves a slightly lower energy in the end.

4.4.3 Motion

Figure 4.14(a) shows one image of a motion sequence where a cat moves against moving background. This is a difficult sequence because the cat's motion is non-rigid. Figures 4.14(b) and (c) show the horizontal and vertical motions detected with our expansion algorithm. Notice that the cat has been accurately localized. Even the tail and parts of the legs are clearly separated from the background motion.

Chapter 5

Piecewise constant prior

In this chapter we develop minimization algorithms for an energy whose smoothness term expresses a piecewise constant prior. The piecewise constant prior can be viewed as an important special case of the piecewise smooth prior, and even in this case the minimization problem is NP-complete. For the special case of a piecewise constant prior, we show that finding the minimum of the energy is equivalent to finding a minimum cost multiway cut on a certain graph (section 5.2). The multiway cut is a generalization of the standard two-terminal graph cut problem. Finding a minimum cost multiway cut is an NP-complete problem for which there are good approximation algorithms. We apply our own swap and expansion algorithms developed in chapter 4 to minimize the energy function in this chapter. The solution given by the expansion algorithm is within a factor of two from the optimal solution. In addition, in section 5.3.2, we develop a new algorithm that finds a local minimum in the jump move space. Even though piecewise constant priors model simpler interactions than piecewise smooth priors, we show that it can be successfully used for problems where the number of labels is not large.

5.1 Preliminaries

The piecewise constant smoothness prior is the simplest smoothness prior that allows discontinuities. It can be modeled by the following neighbor interaction function:

$$V_{\{p,q\}} = u_{\{p,q\}} \cdot \delta(f_p \neq f_q),$$

where

$$\delta(f_p \neq f_q) = \begin{cases} 1 & \text{if } f(p) \neq f(q), \\ 0 & \text{otherwise.} \end{cases}$$

This neighbor interaction function gives penalty $u_{\{p,q\}}$ for assigning different labels to two neighboring pixels. Notice that this penalty does not depend on the labels assigned, as long as they are different. The graph of this function is shown in figure 1.7.

The resulting energy function is:

$$E_P(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} u_{\{p,q\}} \cdot \delta(f_p \neq f_q) \tag{5.1}$$

When $u_{\{p,q\}} = const$, the smoothness energy in 5.1 comes from a particular MRF, and it is called *Potts* energy. We call the smoothness energy in 5.1 the generalized Potts energy. Minimizing $E_P(f)$ is an NP-hard problem, as will be shown in the appendix.

5.2 Multiway cut

We now show that the problem of minimizing the generalized Potts energy $E_P(f)$ can be solved by computing a minimum cost multiway cut on a certain graph.

Consider a graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with non-negative edge weights, along with a set of terminal vertices $\mathcal{L} \subset \mathcal{V}$. A subset of edges $\mathcal{C} \subset \mathcal{E}$ is called a *multiway cut* if the terminals are completely separated in the induced graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$. We will also require that no proper subset of $\mathcal{C}$ separates the terminals in $\mathcal{G}(\mathcal{C})$. The cost of the multiway cut $\mathcal{C}$ is denoted by $|\mathcal{C}|$ and equals the sum of its edge weights. The *multiway cut problem* is to find the minimum cost multiway cut [14]. The multiway cut problem is a generalization of the standard two-terminal graph cut problem.

We take $\mathcal{V} = \mathcal{P} \cup \mathcal{L}$. This means that $\mathcal{G}$ contains two types of vertices: *p-vertices* (pixels) and *l-vertices* (labels). Note that l-vertices will serve as terminals for our multiway cut problem. Two p-vertices are connected by an edge if and only if the corresponding pixels are neighbors in the neighborhood system $\mathcal{N}$. The set $\mathcal{E}_{\mathcal{N}}$ consists of the edges between p-vertices, which we will call *n-links*. Each n-link $\{p, q\} \in \mathcal{E}_{\mathcal{N}}$ is assigned a weight

$$w_{\{p,q\}} = u_{\{p,q\}}. \tag{5.2}$$

Each p-vertex is connected by an edge to each l-vertex. An edge $\{p, l\}$ that connects a p-vertex with a terminal (an l-vertex) will be called a *t-link* and the set of all such edges will be denoted by $\mathcal{E}_{\mathcal{T}}$. Each t-link $\{p, l\} \in \mathcal{E}_{\mathcal{T}}$ is assigned a weight

$$w_{\{p,l\}} = K_p - D_p(l), \tag{5.3}$$

where $K_p > \max_l D_p(l)$ is a constant that is large enough to make the weights positive. The edges of the graph are $\mathcal{E} = \mathcal{E}_{\mathcal{N}} \cup \mathcal{E}_{\mathcal{T}}$. Figure 5.1 shows the structure of the graph $\mathcal{G}$.

We now establish a one-to-one correspondence between multiway cuts and labelings. As an illustration, figure 5.2 shows an induced graph $\mathcal{G}(\mathcal{C}) = \langle \mathcal{V}, \mathcal{E} - \mathcal{C} \rangle$ corresponding to some multiway cut $\mathcal{C}$ on $\mathcal{G}$ given in figure 5.1. It is easy to see that in general there should be exactly one t-link to each p-vertex in the induced

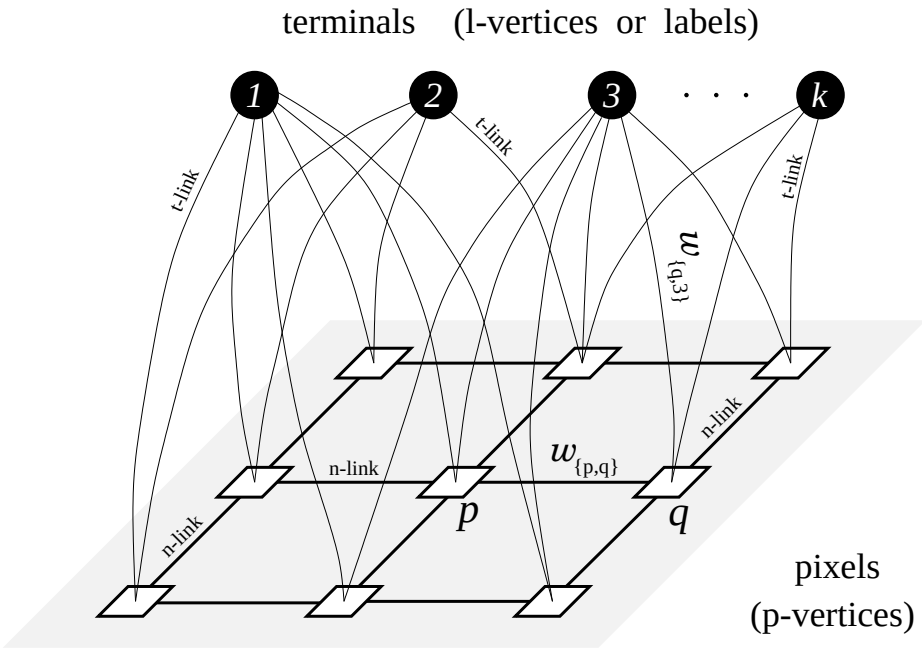


Figure 5.1: An example of the graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with terminals $\mathcal{L} = \{1, \ldots, k\}$. The pixels $p \in \mathcal{P}$ are shown as white squares. Each pixel has an n-link to its four neighbors. Each pixel is also connected to all terminals by t-links (some of the t-links are omitted from the drawing for legibility). The set of vertices $\mathcal{V} = \mathcal{P} \cup \mathcal{L}$ includes all pixels and terminals. The set of edges $\mathcal{E} = \mathcal{E}_{\mathcal{N}} \cup \mathcal{E}_{\mathcal{T}}$ consists of all n-links and t-links.

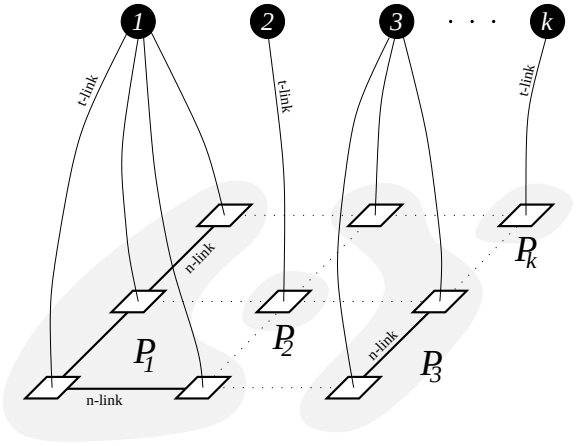


Figure 5.2: Graph induced by a multiway cut

graph $\mathcal{G}(\mathcal{C})$: if there were two or more t-links to a pixel, $\mathcal{C}$ would not separate the corresponding terminals; while if there were no t-links, a proper subset of $\mathcal{C}$ would separate the terminals. This yields an isomorphism between the set of all multiway cuts and the set of all possible labelings f. A multiway cut $\mathcal{C}$ corresponds to the labeling $f^{\mathcal{C}}$ which assigns the label l to all pixels p which are t-linked to the l-vertex in $\mathcal{G}(\mathcal{C})$. It is clear that the n-link between p and q is included in $\mathcal{C}$ just in case $f^{\mathcal{C}}(p) \neq f^{\mathcal{C}}(q)$: if such an n-link were not included, $\mathcal{C}$ would not be a cut; while if the n-link were in $\mathcal{C}$, but $f^{\mathcal{C}}(p) = f^{\mathcal{C}}(q)$, then a proper subset of $\mathcal{C}$ would be a cut.

Theorem 2 *If $\mathcal{C}$ is a multiway cut on $\mathcal{G}$, then $|\mathcal{C}| = E_P(f^{\mathcal{C}})$ plus a constant.*

PROOF: The cost of the cut $\mathcal{C}$ is the sum of the weights of the n-links and the t-links in $\mathcal{C}$. If $f^{\mathcal{C}}(p) \neq f^{\mathcal{C}}(q)$ then a pair of neighboring pixels $\{p, q\}$ contributes $w_{\{p,q\}}$ to the sum of the n-links in $\mathcal{C}$ which is

$$\sum_{\{p,q\} \in \mathcal{E}_{\mathcal{N}}} w_{\{p,q\}} \cdot \delta(f^{\mathcal{C}}(p) \neq f^{\mathcal{C}}(q)). \tag{5.4}$$

The sum of the t-links is

$$\sum_{p \in \mathcal{P}} \sum_{\substack{l \in \mathcal{L} \\ l \neq f^{\mathcal{C}}(p)}} w_{\{p,l\}} = \sum_{p \in \mathcal{P}} \sum_{\substack{l \in \mathcal{L} \\ l \neq f^{\mathcal{C}}(p)}} (K_p - D_p(l)) = \sum_{p \in \mathcal{P}} \sum_{\substack{l \in \mathcal{L} \\ l \neq f^{\mathcal{C}}(p)}} K_p - \sum_{p \in \mathcal{P}} \sum_{\substack{l \in \mathcal{L} \\ l \neq f^{\mathcal{C}}(p)}} D_p(l).$$

This can be re-written as

$$(|\mathcal{L}| - 1) \cdot \sum_{p \in \mathcal{P}} K_p \; - \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}} D_p(l) \; + \sum_{p \in \mathcal{P}} D_p(f^{\mathcal{C}}(p)). \tag{5.5}$$

Only the last term depends on $\mathcal{C}$; the other terms are constants. So by adding the costs in (5.4) and (5.5), we obtain $E_P(f^{\mathcal{C}})$ from (5.1) plus a constant. ■

Corollary 1 *If $\mathcal{C}$ is a minimum cost multiway cut on $\mathcal{G}$, then $f^{\mathcal{C}}$ minimizes E_P.*

When the number of terminals is 2, the multiway cut problem reduces to the standard graph cut problem which can be solved efficiently. Thus when there are just 2 labels, the exact minimum of $E_P(f)$ can be found. This result was first reported by Greig et al. in [21].

While the multiway cut problem is known to be NP-complete if there are more than 2 terminals, there is a fast approximation algorithm [14]. This algorithm works as follows. First, for each terminal $l \in \mathcal{L}$ it finds an *isolating* two-way minimum cut $\mathcal{C}(l)$ that separates l from all other terminals. This is just the standard graph cut problem. Then the algorithm generates a multiway cut $\mathcal{C} = \cup_{l \neq l_{max}} \mathcal{C}(l)$ where $l_{max} = \arg\max_{l \in \mathcal{L}} |\mathcal{C}(l)|$ is the terminal with the largest cost isolating cut. This "isolation heuristic" algorithm produces a cut which is optimal to within a factor of $2 - \frac{2}{|\mathcal{L}|}$. However, the isolation heuristic algorithm suffers from two problems that limits its applicability to our energy minimization problem.

- The algorithm will assign many pixels a label that is chosen essentially arbitrarily. Note that the union of all isolating cuts $\cup_{l \in \mathcal{L}} \mathcal{C}(l)$ may leave some vertices disconnected from any terminal. The multiway cut $\mathcal{C} = \cup_{l \neq l_{max}} \mathcal{C}(l)$ connects all those vertices to the terminal l_{max}.

- While the multiway cut $\mathcal{C}$ produced is close to optimal, this does not imply that the resulting labeling $f^{\mathcal{C}}$ is close to optimal. This is due to the constant that appears in theorem 2; this results from equation (5.3), where we needed to add constants to each t-link to ensure that the weights were positive. As a result, the isolation heuristic algorithm does not produce a labeling whose energy is within a constant factor of optimal.

5.3 Local energy minimization

5.3.1 Swap and expansion move space

The neighbor interaction function $V_{\{p,q\}}(f_p, f_q) = u_{\{p,q\}}\delta(f_p \neq f_q)$ obviously satisfies the assumptions in (4.2) and (4.3). Therefore we can apply the swap and expansion move space algorithms for E_P. Moreover, we can simplify the weights on t-links for the swap move space algorithm, since

$$\sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(\alpha, f_q) = \sum_{\substack{q \in \mathcal{N}_p \\ q \notin S}} V_{\{p,q\}}(\beta, f_q).$$

Recall that $S = \{p \mid f_p \in \{\alpha, \beta\}\}$. The simplified edges are in the table below:

edge	**weight**	**for**
t_p^α	$D_p(\alpha)$	$p \in S$
t_p^β	$D_p(\beta)$	$p \in S$

The expansion move space algorithm gives a solution within a factor of two from the optimal solution. In section 4.3.4 we showed that the expansion algorithm produces an answer within a factor of $2c$ from the optimal, where c is defined by (4.13):

$$c = \max_{\{p,q\} \in \mathcal{N}} \left(\frac{\max_{l_1 \neq l_2 \in \mathcal{L}} V_{\{p,q\}}(l_1, l_2)}{\min_{l_3 \neq l_4 \in \mathcal{L}} V_{\{p,q\}}(l_3, l_4)} \right).$$

For $E_P(f)$

$$\max_{l_1 \neq l_2 \in \mathcal{L}} V_{\{p,q\}}(l_1, l_2) = \min_{l_3 \neq l_4 \in \mathcal{L}} V_{\{p,q\}}(l_3, l_4) = u_{\{p,q\}},$$

and so $c = 1$.

5.3.2 Jump move space

In this section we explain how to find a local minimum of E_P in the jump move space. The structure of the algorithm is in figure 4.2. We just have to explain how to perform step 3.1, that is how to find an optimal jump move.

Recall that for the jump move space there is a function $h : \mathcal{L} \to \{0, 1, \ldots, k-1\}$, where k is the number of labels. The jump move space is indexed by an integer $i \in \{-k+1, -k+2, \ldots, k-1\}$. To simplify the notation, we will assume without loss of generality that $\mathcal{L} = \{0, 1, \ldots, k-1\}$ and h is the identity function. Thus we will refer to $h(l)$ as just l.

Given a labeling f and an integer i, the task is to find an optimal i-jump move from f. Like the previous methods, the technique is based on computing a cut on a certain graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. Its structure will be dynamically determined by a labeling f and an integer i.

This section is organized as follows. First we describe the construction of $\mathcal{G}$ for a given f and i. We show that cuts $\mathcal{C}$ on $\mathcal{G}$ correspond in a natural way to labelings $f^{\mathcal{C}}$ which are within one i-jump move of f. Then, based on a number of simple properties, we define a class of *elementary* cuts. Theorem 3 shows that elementary cuts are in one to one correspondence with the set of labelings that are within one i-jump of f, and also that the cost of an elementary cut is $|\mathcal{C}| = E_P(f^{\mathcal{C}})$ plus a constant. A corollary from this theorem states our main result that the desired labeling $\hat{f}$ equals $f^{\mathcal{C}}$ where $\mathcal{C}$ is a minimum cut on $\mathcal{G}$.

The structure of the graph is illustrated in figure 5.4. The set of vertices $\mathcal{V}$ contains two terminals 0 and 1, and all pixels $p \in \mathcal{P}$. In addition, for each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $|f_p - f_q| = |i|$, we create an auxiliary vertex $a_{\{p,q\}}$. Thus, the set of vertices is

$$\mathcal{V} \;=\; \{0\} \cup \{1\} \cup \mathcal{P} \cup \left(\bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ |f_p - f_q| = |i|}} a_{\{p,q\}} \right).$$

Before describing the rest of the graph, we need a few definitions. Recall that in the algorithm for the swap move space, the label assigned to a pixel p depends only on the terminal from which p is disconnected: if p is disconnected from terminal α, it is assigned label α and if p is disconnected from the terminal β, it is assigned label β. In the algorithm for the expansion move space the label assigned to pixel p does not depend on p if p is disconnected from α. However the label does depend on p if p is disconnected from $\bar{\alpha}$; i.e. in this case p is assigned its old label f_p. In the algorithm for the jump move space, the label assigned to a pixel will always depend on this pixel no matter which terminal it is disconnected from. We construct two sets, $\mathcal{L}_1$ and $\mathcal{L}_0$, such that if p is disconnected from terminal 1, it will be assigned a certain $l \in \mathcal{L}_1$; and if p is disconnected from terminal 0, it will be assigned a certain $l \in \mathcal{L}_0$.

We design $\mathcal{L}_0$ and $\mathcal{L}_1$ so that $\mathcal{L} \cup \{-1\} = \mathcal{L}_0 \bigcup \mathcal{L}_1$. Here -1 is a special symbol which corresponds to a dummy label. We ensure that $\mathcal{L}_0 \cap \mathcal{L}_1 = \{-1\}$, and so $\mathcal{L}_0 - \{-1\}$ and $\mathcal{L}_1 - \{-1\}$ split $\mathcal{L}$ into two disjoint sets.

To define $\mathcal{L}_0$ and $\mathcal{L}_1$ we introduce a binary function $loc : \mathcal{L} \to \{0, 1\}$ defined as follows. For $l \in \mathcal{L}$ there exists the unique decomposition $l = qi + r$ where r and q are integers such that $r < i$.

$$loc(l) \;=\; \begin{cases} 0 & \text{if } \; q \text{ is even} \\ 1 & \text{otherwise.} \end{cases}$$

Having defined the function loc,

$$\mathcal{L}_0 = \{l \mid loc(l) = 0\} \cup \{-1\}$$

and

$$\mathcal{L}_1 = \{l \mid loc(l) = 1\} \cup \{-1\}.$$

Lemma 8 *Let $l \in \mathcal{L}$. If $0 \leq l + i \leq k - 1$, then $loc(l) \neq loc(l + i)$.*

PROOF: Let $l = qi + r$ be the unique decomposition of l where r and q are integers such that $r < i$. Then $l + i = (q + 1)i + r$ is the unique decomposition for $l + i$. Obviously one of q and $q + 1$ is odd and the other one is even, and so we have the required result. ■

Now we define the functions $g_0 : \mathcal{L} \to \mathcal{L} \cup \{-1\}$ and $g_1 : \mathcal{L} \to \mathcal{L} \cup \{-1\}$. If p is disconnected from terminal 0, p is assigned label $g_0(f_p)$, and if p is disconnected from terminal 1, it is assigned $g_1(f_p)$.

$$g_0(f_p) = \begin{cases} f_p & \text{if } f_p \in \mathcal{L}_0 \\ f_p + i & \text{if } f_p + i \in \mathcal{L}_0 \\ -1 & \text{otherwise.} \end{cases}$$

Similarly,

$$g_1(f_p) = \begin{cases} f_p & \text{if } f_p \in \mathcal{L}_1 \\ f_p + i & \text{if } f_p + i \in \mathcal{L}_1 \\ -1 & \text{otherwise.} \end{cases}$$

Notice that due to Lemma 8, g_0 and g_1 are well defined.

Now we can describe the rest of the graph. Each pixel $p \in \mathcal{P}$ is connected to the terminals 0 and 1 by t-links t_p^0 and t_p^1, correspondingly. Each pair of neighboring pixels $\{p, q\} \in \mathcal{N}$ such that $f_p = f_q$ is connected by an n-link $e_{\{p,q\}}$. For each $\{p, q\} \in \mathcal{N}$ such that $|f_p - f_q| = |i|$ we create a triplet of edges

$$\mathcal{E}_{\{p,q\}} = \left\{ e_{\{p,a\}}, e_{\{a,q\}}, t_a \right\}$$

t_a **is connected to**	**conditions**
1	$f_p = f_q + i$ and $loc(f_p) = 0$
0	$f_p = f_q + i$ and $loc(f_p) = 1$
1	$f_q = f_p + i$ and $loc(f_q) = 0$
0	$f_q = f_p + i$ and $loc(f_q) = 1$

Figure 5.3: Table of connections for edge t_a

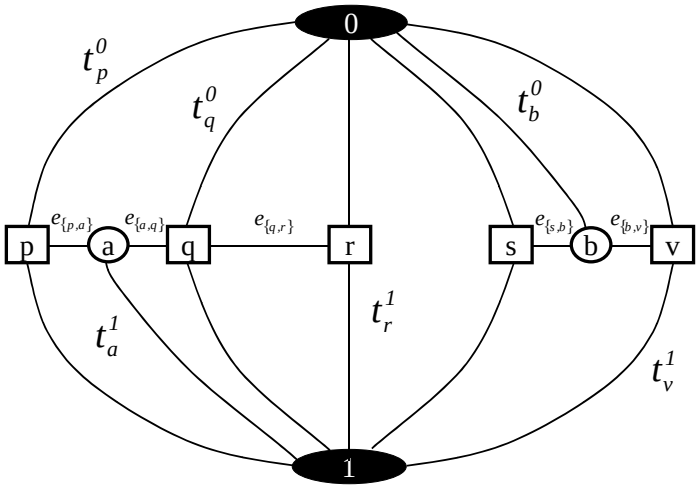


Figure 5.4: An example of the graph $\mathcal{G}$ for a 1D image. The set of pixels in the image is $\mathcal{P} = \{p, q, r, s, v\}$. The neighborhood system is $\mathcal{N} = \{\{p, q\}, \{q, r\}, \{r, s\}, \{s, v\}\}$. Also assume that $|f_p - f_q| = |i|$ and $|f_s - f_v| = |i|$, therefore we create two auxiliary nodes a and b between neighbor pairs $\{p, q\}$ and $\{s, v\}$. $f_q = f_r$ so there is an n-link between q and r. Since $f_r \neq f_s$ and $|f_r - f_s| \neq |i|$ there is neither an n-link nor auxiliary pixel construction between pixels r and s.

where $a = a_{\{p,q\}}$ is the corresponding auxiliary node. The n-links $e_{\{p,a\}}$ and $e_{\{a,q\}}$ connect pixels p and q to $a_{\{p,q\}}$ and the t-link t_a connect the auxiliary node $a_{\{p,q\}}$ to terminal 0 or 1 as summarized in the table in figure 5.3.2.

Finally, we can write the set of all edges as

$$\mathcal{E} \;=\; \left(\bigcup_{p\in\mathcal{P}} \{t_p^{\alpha}, t_p^{\bar{\alpha}}\}\right) \;\cup\; \left(\bigcup_{\substack{\{p,q\}\in\mathcal{N}\\ |f_p - f_q| = |i|}} \mathcal{E}_{\{p,q\}}\right) \;\cup\; \left(\bigcup_{\substack{\{p,q\}\in\mathcal{N}\\ f_p = f_q}} e_{\{p,q\}}\right).$$

The weights assigned to the edges are shown in the table below. Here $S = \{p \mid 0 \le f_p + i \le k-1\}$. That is S is the set of pixels with labels which can be increased by i and still be in the legal range. We define $D_p(-1) = \infty$, that is the cost of assigning a dummy label is infinitely high.

edge	**weight**	**for all**
t_p^0	$D_p(g_0(f_p))$	$p \in \mathcal{P}$
t_p^1	$D_p(g_1(f_p))$	
$e_{\{p,a\}},\ e_{\{a,q\}},$ and t_a	$u_{\{p,q\}}$	$\{p,q\} \in \mathcal{N},\ \lvert f_p - f_q\rvert = \lvert i\rvert$
$e_{\{p,q\}}$	$u_{\{p,q\}}$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Any cut $\mathcal{C}$ on the graph $\mathcal{G}$ must sever (include) exactly one t-link for any pixel $p \in \mathcal{P}$, as proved in lemma 2. This defines a natural labeling $f^{\mathcal{C}}$ corresponding to a cut $\mathcal{C}$ on $\mathcal{G}$. Formally,

$$f_p^{\mathcal{C}} = \begin{cases} g_0(f_p) & \text{if} \quad t_p^0 \in \mathcal{C} \\ g_1(f_p) & \text{if} \quad t_p^1 \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P}. \tag{5.6}$$

Lemma 9 *A cut $\mathcal{C}$ on $\mathcal{G}$ corresponds to a labeling $f^{\mathcal{C}}$ which is one i-jump away from the original labeling f.*

PROOF: $g_0(f_p)$ and $g_1(f_p)$ are defined to be f_p, $f_p + i$, or -1. Due to the infinite penalties for assigning label -1, p will be assigned either f_p or $f_p + i$ in the labeling $f^{\mathcal{C}}$, which means $(f, f^{\mathcal{C}})$ is an i-jump. ■

We state property 4 and lemma 10 without a proof since their proofs are very similar to property 2 and lemma 6.

Property 4 *Let p and q be neighboring pixels such that $f_p = f_q$. For any cut $\mathcal{C}$ and for any n-link $e_{\{p,q\}}$:*

$$\begin{aligned} &a) \quad \textit{If} \quad t_p^0, t_q^0 \in \mathcal{C} \quad \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ &b) \quad \textit{If} \quad t_p^1, t_q^1 \in \mathcal{C} \quad \textit{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\ &c) \quad \textit{If} \quad t_p^0, t_q^1 \in \mathcal{C} \quad \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \\ &d) \quad \textit{If} \quad t_p^1, t_q^0 \in \mathcal{C} \quad \textit{then} \quad e_{\{p,q\}} \in \mathcal{C}. \end{aligned}$$

Lemma 10 *If $\{p,q\} \in \mathcal{N}$ and $f_p = f_q$ then any cut $\mathcal{C}$ on $\mathcal{G}$ satisfies*

$$|\mathcal{C} \cap e_{\{p,q\}}| = V_{p,q}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}).$$

We state properties 5 and 6 without proofs; the proofs are similar to property 3.

Property 5 *Let p and q be such that $|f_p - f_q| = |i|$, and $a = a_{\{p,q\}}$ be the auxiliary pixel between p and q. If t_a connects a to terminal 0, then a minimum cut cut $\mathcal{C}$ on $\mathcal{G}$ satisfies:*

$$\begin{array}{llll} a) & \textit{If} & t_p^0, t_q^0 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a. \\ b) & \textit{If} & t_p^1, t_q^1 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ c) & \textit{If} & t_p^0, t_q^1 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \\ d) & \textit{If} & t_p^1, t_q^0 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \end{array}$$

Property 6 *Let p and q be such that $|f_p - f_q| = |i|$, and $a = a_{\{p,q\}}$ be the auxiliary pixel between p and q. If t_a connects a to terminal 1, then a minimum cut cut $\mathcal{C}$ on $\mathcal{G}$ satisfies:*

$$\begin{array}{llll} a) & \textit{If} & t_p^0, t_q^0 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ b) & \textit{If} & t_p^1, t_q^1 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a. \\ c) & \textit{If} & t_p^0, t_q^1 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \\ d) & \textit{If} & t_p^1, t_q^0 \in \mathcal{C} & \textit{then} \quad \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \end{array}$$

Lemma 11 follows from figure 5.3.2, properties 5, 6 and is proven similar to lemma 7.

Lemma 11 *If $\{p,q\} \in \mathcal{N}$ and $|f_p - f_q| = |i|$ then the minimum cut $\mathcal{C}$ on $\mathcal{G}_\alpha$ satisfies*

$$|\mathcal{C} \cap \mathcal{E}_{\{p,q\}}| = V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}})$$

Property 4 holds for any cut, and properties 5 and 6 holds for a minimum cut. However, there can be other cuts besides the minimum cut that satisfy these two properties. We will define an *elementary* cut on $\mathcal{G}$ to be a cut that satisfies properties 4, 5, and 6.

Theorem 3 *There is a one to one correspondence between the set of all elementary cuts on $\mathcal{G}$ and the set of all labelings within one i-jump of f. Moreover, for any elementary cut $\mathcal{C}$ we have $|\mathcal{C}| = E_P(f^{\mathcal{C}})$ plus a constant.*

PROOF: Similarly to the proof of theorem 2, one to one correspondence between the set of all elementary cuts and the set of all labelings within one i-jump of f follows from properties 4, 5, and 6.

We now compute the cost of a elementary cut $\mathcal{C}$, which is

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}} |\mathcal{C} \cap \{t_p^0, t_p^1\}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} |\mathcal{C} \cap e_{\{p,q\}}| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ |f_p - f_q| = |i|}} |\mathcal{C} \cap \mathcal{E}_{\{p,q\}}|. \quad (5.7)$$

$$|\mathcal{C} \cap \{t_p^0, t_p^1\}| = \begin{cases} D_p(g_0(f_p)) & \text{if } t_p^0 \in \mathcal{C} \\ D_p(g_1(f_p)) & \text{if } t_p^1 \in \mathcal{C} \end{cases} = D_p(f_p^{\mathcal{C}}).$$

Therefore, the first term in (5.7) is

$$\sum_p |\mathcal{C} \cap \{t_p^0, t_p^1\}| = \sum_p D_p(f_p^{\mathcal{C}}).$$

Lemmas 10 and 11 hold for elementary cuts, since they were based on properties 4, 5, and 6. Then, lemmas 10 and 11 give us the second and the third terms in (5.7). Thus, the total cost of a elementary cut $\mathcal{C}$ is

$$|\mathcal{C}| = \sum_p D_p(f_p^{\mathcal{C}}) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ |f_p - f_q| = |i|}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) =$$

$$= \sum_p D_p(f_p^{\mathcal{C}}) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) - \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q \\ |f_p - f_q| \neq |i|}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) =$$

$$= \sum_p D_p(f_p^{\mathcal{C}}) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p^{\mathcal{C}}, f_q^{\mathcal{C}}) - \sum_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q \\ |f_p - f_q| \neq |i|}} u_{\{p,q\}} =$$

$$= E_P(f^{\mathcal{C}}) + const$$

■

Our main result is a simple consequence of this theorem, since the minimum cut is an elementary cut.

Corollary 3 *Let $\mathcal{C}$ be the minimum cut on $\mathcal{G}$. Then the optimal labeling $\hat{f} = \arg\min E_P(f')$ among f' within one i-jump of f is given by $\hat{f} = f^{\mathcal{C}}$.*

5.4 Experimental results

In this section we present experimental results for image restoration and stereo. To assess how well our algorithms perform energy minimization, we compare the energy achieved by our algorithms to the energy produced by simulated annealing. To assess how well our algorithms solve the visual correspondence problem, we compare the results to normalized correlation, the standard visual correspondence

method. Since we found the performance of our swap, jump, and expansion algorithms to be very similar, we show the results only for the expansion algorithm everywhere except for the real stereo pair with the ground truth in section 5.4.2. For this stereo pair we perform extensive comparisons of the swap, jump, and expansion move space algorithms against each other.

For our method the coefficients $u_{\{p,q\}}$'s vary as proposed in section 3.3.1. In particular, we chose

$$u_{\{p,q\}} = \begin{cases} \lambda & \text{if } |I_p - I_q| < 5 \\ 2\lambda & \text{otherwise} \end{cases}$$

$D_p(f_p)$'s are chosen as explained in 2.4.2. For normalized correlation we chose the parameters which give the best statistics when the ground truth is available (i.e. minimizing the total number of errors); when the ground truth is not available we chose the parameters which appear to give the best results. For simulated annealing we used the "Metropolis-Heat bath" version with truncated logarithmic cooling schedules. To give it a good starting point, simulated annealing was initialized with the input image for the image restoration problem and with the results of normalized correlation for the visual correspondence problem. In contrast it appears empirically that the initial labeling for our method is not important. The speed improves by about 1.5 when we initialize with a good starting point, but the final answers differ by less than 2% of pixels for any starting point we have tried.

5.4.1 Image restoration

Example 1

In this section we test the expansion move space algorithm on the image restoration problem shown in figure 3.3. Notice that this image consists of several regions of constant intensity, and thus it is appropriate for piecewise constant restoration.

The comparison with simulated annealing for $\lambda = 40$ is shown in figure 5.5. Starting from the initial energy of about 800,000 our method converges to the energy of 201,860 in 90 seconds. Most of the progress occurs in the first 30 seconds. In 120 seconds simulated annealing reduces the same initial energy to the energy of about 406,000, and from this point on it makes very little progress in the 15 hours that we ran it. Even though our algorithm finds a local minimum, its value is two times better than what simulated annealing finds in 15 hours. Clearly if speed is important, then our algorithm is superior to simulated annealing for this restricted class of energy functions.

To assess how well our algorithm solves the image restoration problem we compare the labeling computed by our method with the labeling of the original scene. Figure 5.6 shows the restored images for several values of λ. For $\lambda = 0.1$ the restored image is fairly close to the original image. For $\lambda = 3$ the restored image is almost exactly the same as the original image. For $\lambda = 20000$ the restored image

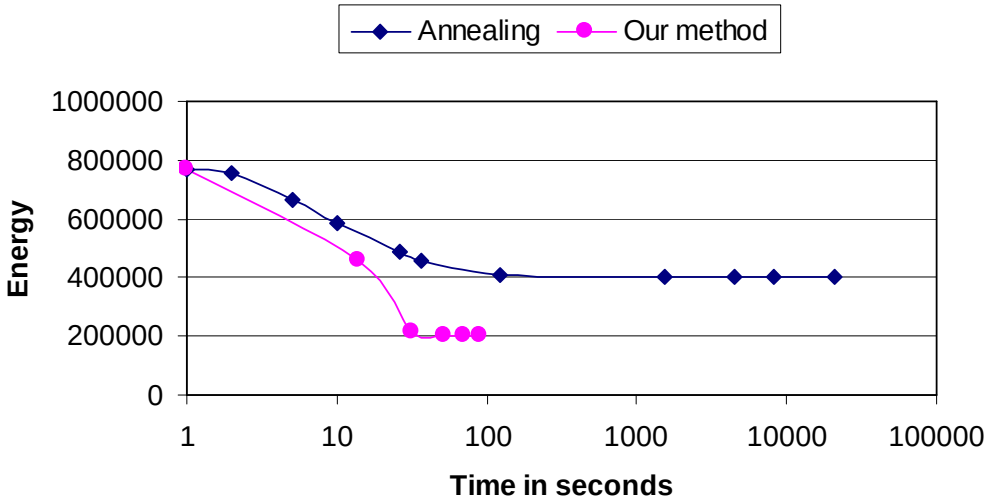
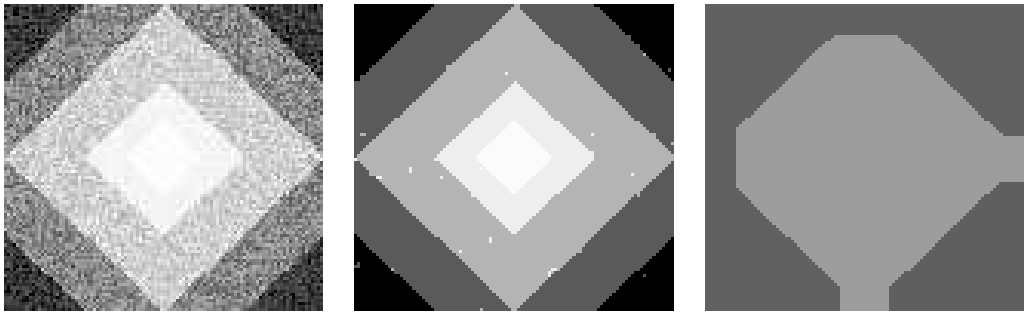


Figure 5.5: Performance comparison on the image restoration task. The data points for our method correspond to the 4 cycles the algorithm takes until convergence.



(a) $\lambda = 0.1$ (b) $\lambda = 3$ (c) $\lambda = 20000$

Figure 5.6: Restored images for various values of λ.

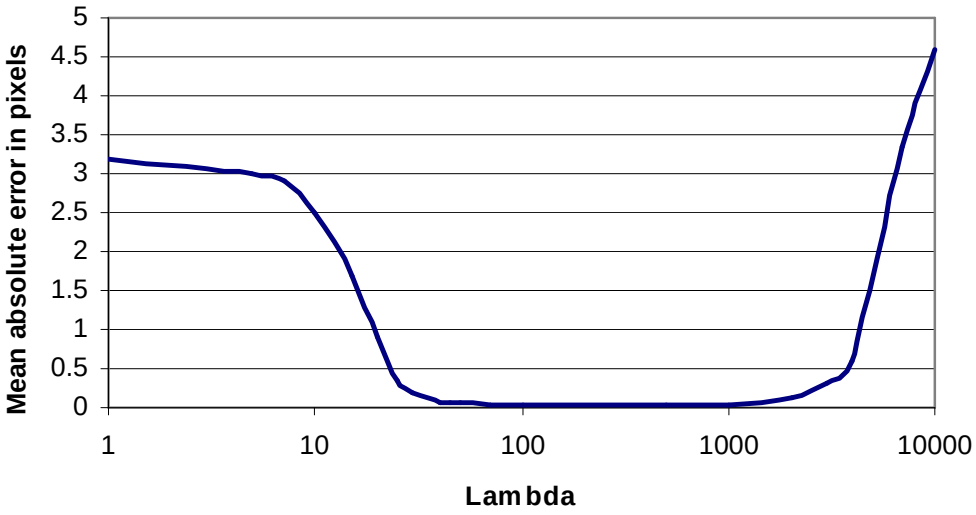


Figure 5.7: Mean absolute error.

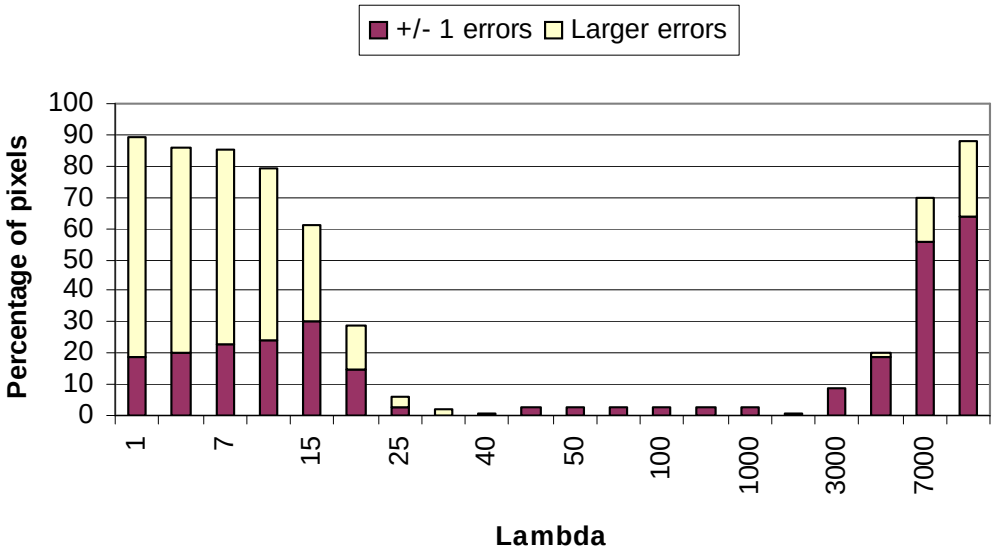
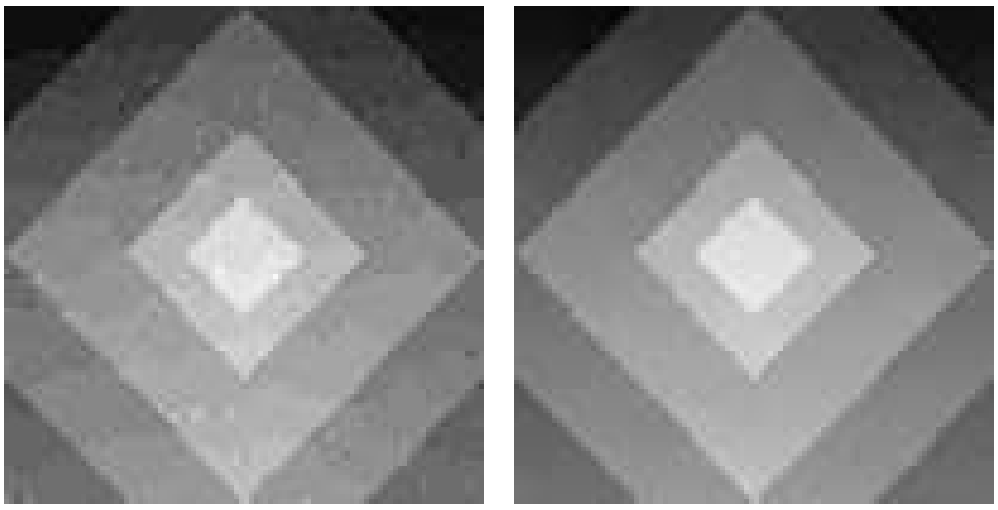


Figure 5.8: Percentage of pixels with nonzero error.



(a)Piecewise constant prior (b) Piecewise smooth prior

Figure 5.9: Images restored with piecewise constant and piecewise smooth priors.

has almost no discontinuities. Figure 5.7 plots mean absolute error versus λ, and figure 5.8 plots the percentage of absolute errors versus λ. As expected, for small values of λ the algorithm performs poorly because it over emphasizes the data. For large values of λ the algorithm performs poorly due to over-smoothing. However for a large range of λ the algorithm finds the original labeling almost exactly. This shows that our algorithm is robust in the choice of parameters.

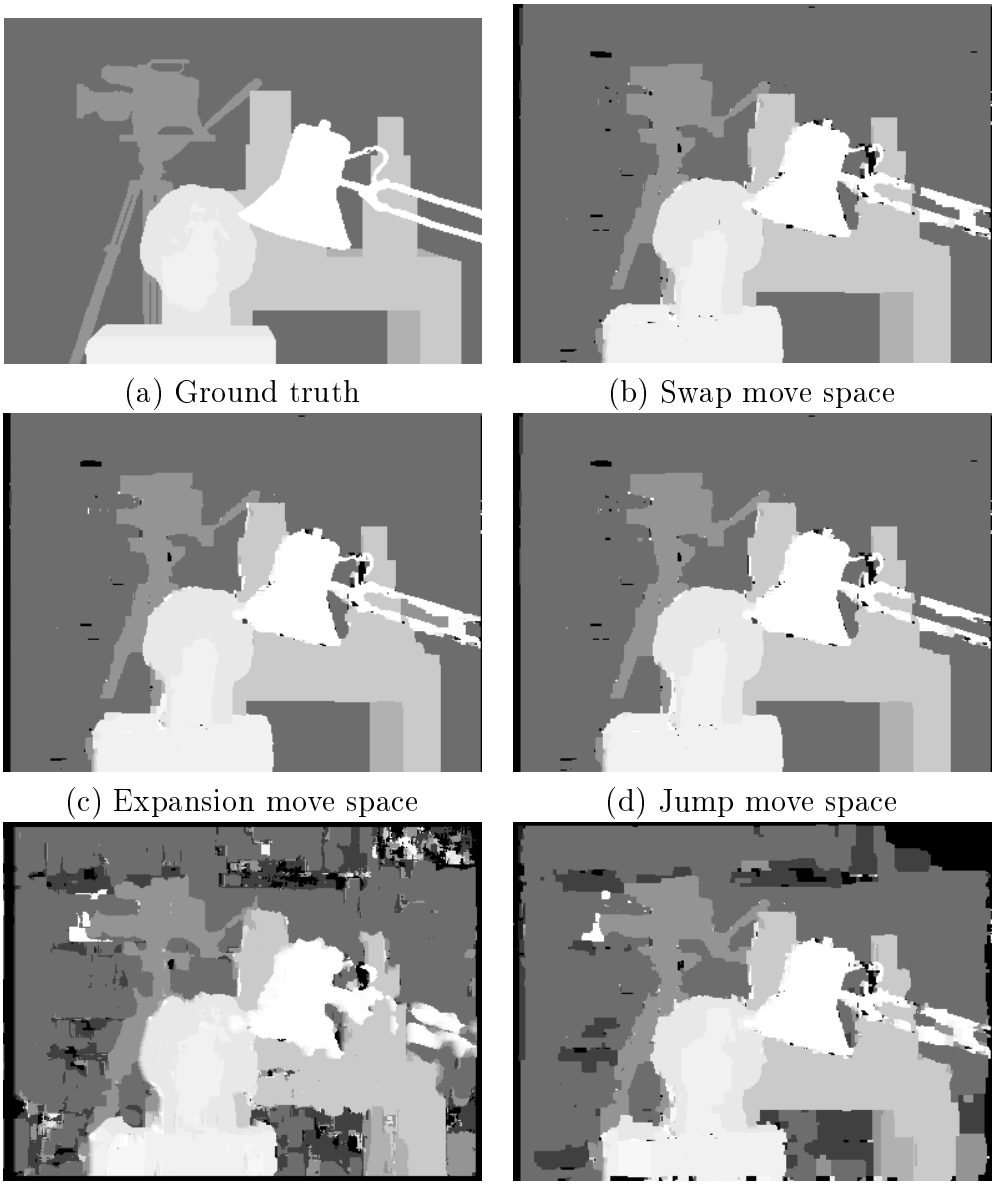
Example 2

In this section we test our algorithm on the image restoration problem shown in figure 4.8. Notice that in this case, the image consists of several regions with intensity smoothly varying inside each region. A piecewise constant prior is not appropriate for this problem. The results of our algorithm are shown in figure 5.9, along with the results from section 4.4.1 where we applied our algorithm for a piecewise smooth prior to this example. As expected, the piecewise constant restoration does not work as well as piecewise smooth restoration; there is a significant "banding" effect if figure 5.9(a) compared to figure 5.9(b).

5.4.2 Stereo

Real imagery with ground truth

The left image of the real stereo pair with known ground truth is shown in figure 3.8(a). Figure 5.10(a) shows the ground truth for this stereo pair. Figures 5.10(b),(c), and (d) show the results of swap, expansion, and jump algorithms



(a) Ground truth (b) Swap move space

(c) Expansion move space (d) Jump move space

(e) Normalized correlation (f) Simulated annealing

Figure 5.10: Real imagery with ground truth

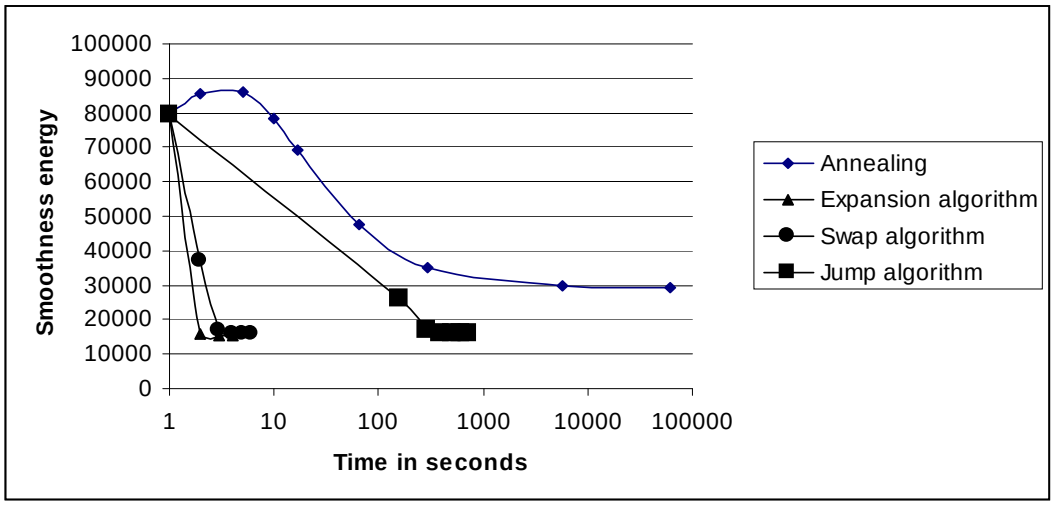


Figure 5.11: Performance comparison of expansion, swap, and jump algorithms with simulated annealing for the problem in figure 3.8(a).

for $\lambda = 20$. Figures 5.10(e) and (f) show the results of normalized correlation and simulated annealing.

The table below summarizes the errors made by the algorithms. In approximately 20 minutes simulated annealing reduces the total errors normalized correlation makes by about one fifth and it cuts the number of ± 1 errors in half. It makes very little additional progress in the rest of 19 hours that we ran it. Our expansion, swap, and jump algorithms make approximately 3 times fewer ± 1 errors and approximately 5 times fewer total errors compared to normalized correlation.

All of our algorithms perform approximately the same. The observed slight difference in errors is quite insignificant (less than one percent). At each cycle the order of labels to iterate over is chosen in a random manner. Another run of the algorithms might give slightly different results where expansion algorithm might do better than the other two algorithms. In general we observed very slight variation between different runs of an algorithm. However the difference in the running time is significant. On average the expansion algorithm takes 3 times less to converge than the swap algorithm and 7 times less to converge than the jump algorithm.

algorithm	% **total errors**	% **of errors** $> \pm 1$	**running time**
expansion algorithm	7.6	2.1	106 sec
swap algorithm	7.0	2.0	300 sec
jump algorithm	7.0	2.0	734 sec
simulated annealing	20.3	5.0	1200 sec
normalized correlation	24.7	10.0	5 sec

Figure 5.11 shows the graph of E_{smooth} versus time for our algorithms versus simulated annealing. Notice that the time axis is on the logarithmic scale. We do

λ	% of total errors	%of errors $> \pm 1$	Absolute average error
1	26.6	4.5	0.40
5	13.0	4.5	0.27
10	7.0	2.3	0.15
20	7.6	2.1	0.15
30	7.9	2.3	0.17
50	8.8	2.3	0.18
100	10.4	2.9	0.21
500	16.3	8.2	0.37

Figure 5.12: Table of errors for the expansion algorithm for different values of λ.

not show the graph for E_{data} because the difference in the E_{data} term all algorithms achieve is insignificant, as expected from the following argument. Most pixels in real images have nearby pixels with very similar intensities. Thus for most pixels p there are a few disparities d for which $D_p(d)$ is approximately the same and small. For the rest of d's, $D_p(d)$ is quite large. This latter group of disparities are essentially excluded from consideration by energy minimizing algorithms. The remaining choices of d are more or less equally likely. Thus the E_{data} term of the energy function has very similar values for our methods and simulated annealing. Our methods quickly reduce the smoothness energy to around $16,000$, while the best simulated annealing can produce in 19 hours is around $30,000$, nearly twice as bad.

The expansion algorithm gives a convergence curve significantly steeper than the other curves; in fact the expansion algorithm makes 99% of the progress in the first iteration. The jump algorithm reduces the energy most slowly out of all our algorithms.

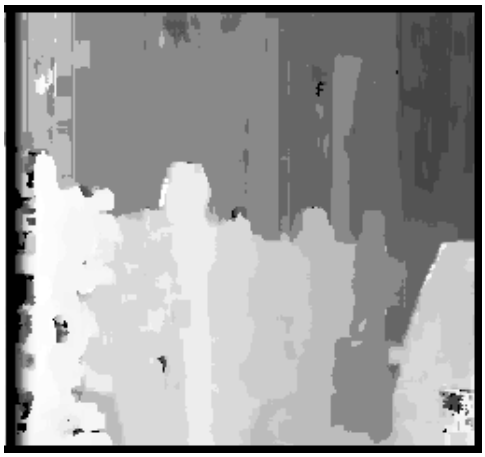
The algorithms appear to be quite stable in the choice of parameter λ. For example the table in figure 5.12 gives the errors made by the expansion algorithm for different choices of λ. For small λ the algorithm makes a lot of errors because it overemphasizes the data, for large values of λ the algorithm makes a lot of errors because it overemphasizes the prior. However for a large interval of λ values the results are good.

Overall there does not appear to be a significant difference in the results given by our algorithms, but the expansion algorithm converges significantly faster than the other algorithms.

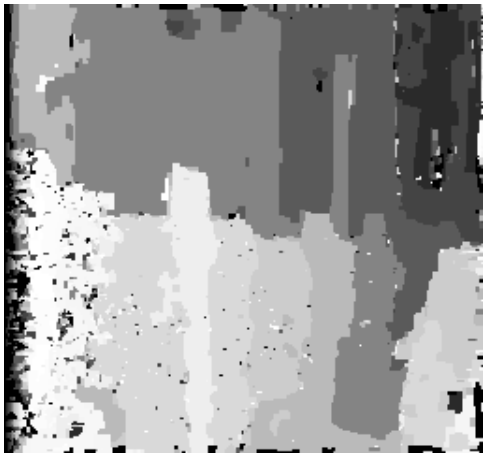
Other real imagery

Figures 5.13 and 5.14 show the results on other standard benchmark image pairs. Simulating annealing was run until it was making very slow progress. Our method was run for two cycles. We show the running times for all algorithms and the final energies for our algorithm and simulated annealing.

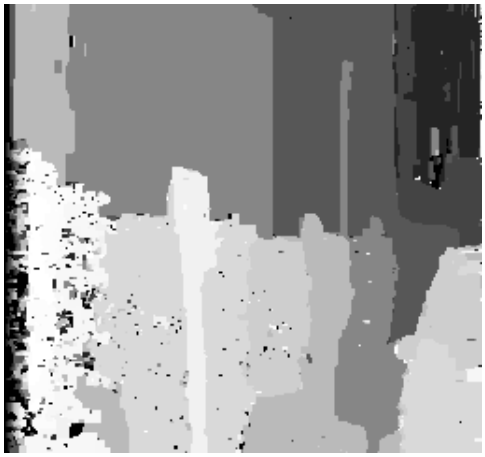
(a) Left image



(b) Normalized correlation: 2 seconds



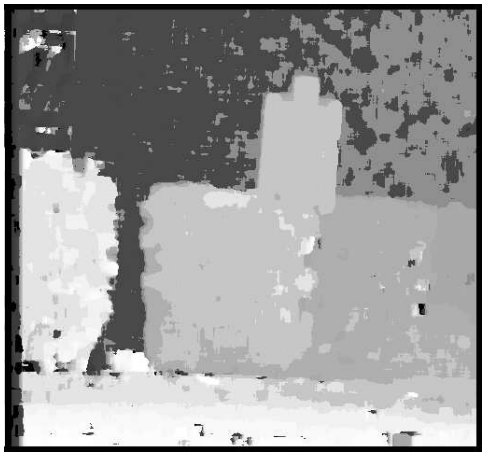
(c) Annealing: $E_{data} = 211,508$, $E_{smooth} = 87,215$; 530 sec



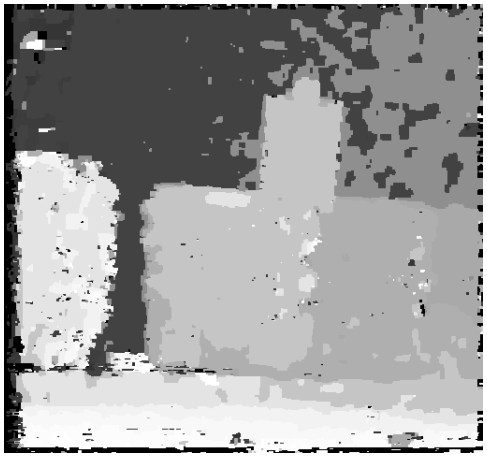
(d) Our result: $E_{data} = 207,672$, $E_{smooth} = 66,680$; 55 seconds

Figure 5.13: The CMU meter image

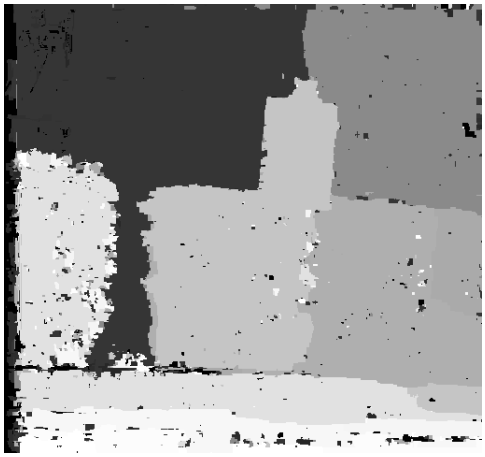
(a) Left image



(b) Normalized correlation: 15 seconds

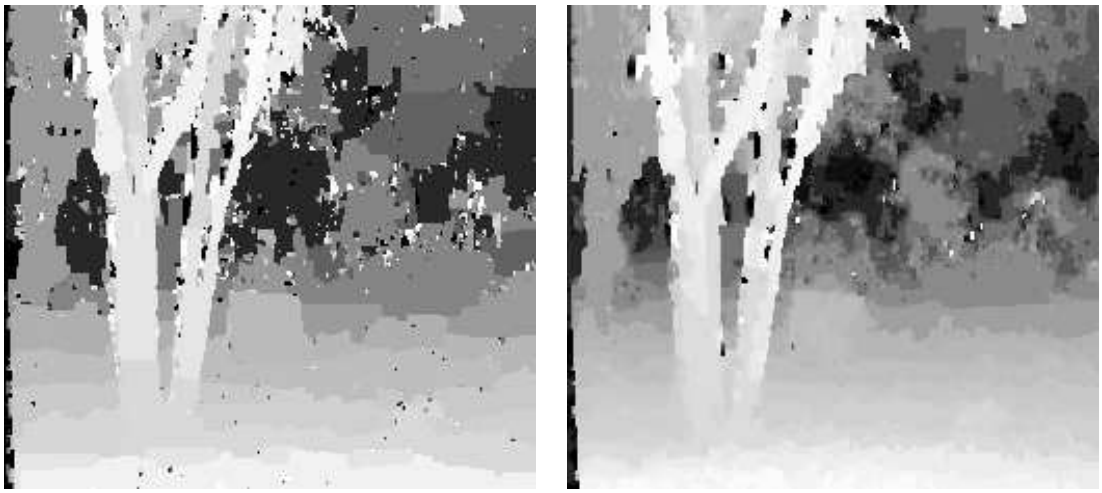


(c) Annealing: $E_{data} = 1,010,140$, $E_{smooth} = 1,137,840$; 7745 sec



(d) Our result: $E_{data} = 874,620$, $E_{smooth} = 528,270$; 577 seconds

Figure 5.14: The shrub image



(a)Piecewise constant prior (b) Piecewise smooth prior

Figure 5.15: Results with piecewise constant and piecewise smooth priors.

Our method correctly localized many details in the images. Examples include the front parking meter and the car in figure 5.13, and the sign in the figure 5.14. Even fine details such as the thin pole on the right in figure 5.13 can be discerned in the output.

Stereo pairs in figures 5.13 and 5.14 do not have many disparities. However if the number of disparities is larger, E_P does not do as well as E_N or E_M which were discussed in chapter 4. For example figure 5.15 compares the results E_P and E_M give for the stereo pair in figure 4.12. Notice that there are fewer disparities found in figure 5.15, since the piecewise constant prior tends to produce large regions with the same disparity.

Appendix

In this appendix we show that minimizing the Potts energy $E_P(f)$ in (5.1) is an NP-complete problem. This also implies that minimizing $E_N(f)$ and $E_M(f)$ in chapter 4 are NP-complete problems.

In section 5.2 we showed that the problem of minimizing the energy $E_P(f)$ in (5.1) over all possible labelings f can be solved by computing a minimum multiway cut on a certain graph. In this appendix we make the reduction in the opposite direction. Specifically, for an arbitrary fixed graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ we will construct an instance of minimizing $E_P(f)$ where the optimal labeling f^* determines a minimum multiway cut on $\mathcal{G}$. This will prove that a polynomial-time method for finding f^* would provide a polynomial-time algorithm for finding the minimum cost multiway cut, which is known to be NP-hard [14]. This NP-hardness proof is based on a construction due to Jon Kleinberg.

The energy minimization problem we address takes as input a set of pixels $\mathcal{P}$, a neighborhood relation $\mathcal{N}$ and a label set $\mathcal{L}$, as well as a set of weights $u_{\{p,q\}}$ and a function $D_p(l)$. The problem is to find the labeling f^* that minimizes the energy $E_P(f)$ given in equation (5.1).

Let $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ be an arbitrary weighted graph with terminal vertices $\{t_1, \ldots, t_k\} \subset \mathcal{V}$ and edge weights $w_{\{p,q\}}$. We will do the energy minimization using $\mathcal{P} = \mathcal{V}$, $\mathcal{N} = \mathcal{E}$, and $u_{\{p,q\}} = w_{\{p,q\}}$. The label set will be $\mathcal{L} = \{1, \ldots, k\}$. Let K be a constant such that $K > E_P(f^*)$; for example, we can select K to be the sum of all $w_{\{p,q\}}$. Our function $D_p(l)$ will force $f^*(t_j) = j$; if $p = t_j$ is a terminal vertex,

$$D_p(l) = \begin{cases} 0 & l = j, \\ K & \text{otherwise.} \end{cases}$$

For a non-terminal vertex p all labels are equally good,

$$\forall l \quad D_p(l) = 0.$$

We will define a labeling f to be *feasible* if the set of pixels labeled j by f forms a connected component that includes t_j. Feasible labelings obviously correspond one-to-one with multiway cuts.

Theorem 3 *The labeling f^* is feasible, and the cost of a feasible labeling is the cost of the corresponding multiway cut.*

PROOF: To prove that f^* is feasible, suppose that there were a set S of pixels that f^* labeled j which were not part of the component containing t_j. We could then obtain a labeling with lower energy by switching this set to the label of some pixel on the boundary of S. The energy of a feasible labeling f is

$$\sum_{\{p,q\}\in\mathcal{N}} u_{\{p,q\}} \cdot \delta(f(p) \neq f(q)),$$

which is the cost of the multiway cut corresponding to f. ■

This shows that minimizing the $E_P(f)$ on an arbitrary $\mathcal{P}$ and $\mathcal{N}$ is intractable. In computer vision, however, $\mathcal{P}$ is usually a planar grid, and combinatorial problems that are intractable on arbitrary graphs sometimes become tractable on the plane or grid.

We now sketch a proof that the energy minimization problem is intractable even when restricted to a planar grid. The reduction is from a special case of the multiway cut problem, where $\mathcal{G}$ is a planar graph with degree 11 and all the edges have weight 1, which is shown to be NP-hard in [14]. We first must embed $\mathcal{G}$ in a grid of pixels, which happens in two stages. In the first stage we convert $\mathcal{G}$ into a planar graph of degree 4. In the second stage we embed this graph in the grid by using a method given in [26]. This embedding can be done in polynomial time; after it is done, each vertex $v \in \mathcal{G}$ corresponds to a connected set of pixels $S(v)$ in the grid, and the adjacency relationships among vertices in $\mathcal{G}$ has been preserved.

The proof now proceeds along the same lines as theorem 3, except for three subtleties. First, we need to ensure that for every vertex v all pixels in $S(v)$ are given the same label. We address this by making the edge weights K between adjacent pixels in $S(v)$. Second, when we embed $\mathcal{G}$ in the grid, there will be gaps. We can solve this by adding additional "grid pixels", which D forces to have the extra label 0 (D will prevent non-grid pixels from having label 0 by making $D_p(0) = K$) and by taking the edge weights between grid pixels and non-grid pixels to be one. The cost of a feasible labeling will be the cost of the corresponding multiway cut plus a constant. Third, the constant $K > E_P(f^*)$ must be now chosen more carefully.

Bibliography

[1] K. Ahuja, Thomas L. Magnati, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] A. Amini, T. Weymouth, and R. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analsis and Machine Intelligence*, 12(9):855–867, 1990.

[3] S.T. Barnard. Stochastic stereo matching over scale. *International Journal of Computer Vision*, 3:17–32, 1989.

[4] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1976.

[5] J. Besag. On the statistical analysis of dirty pictures (with discussion). *Journal of the Royal Statistical Society, Series B*, 48(3):259–302, 1986.

[6] Stan Birchfield and Carlo Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analsis and Machine Intelligence*, 20(4):401–406, 1998.

[7] A. Blake. Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(1):2–12, 1989.

[8] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, MA, 1987.

[9] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–655, 1998.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *International Conference on Computer Vision*, 1999.

[11] Y. Boykov, O. Veksler, and R. Zabih. A new algorithm for energy minimization with discontinuities. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 1999.

[12] V. Cerny. A thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Preprint, Institute of Physics and Biophisics*, 1982.

[13] P.B. Chou and C.M. Brown. The theory and practice of bayesian image labeling. *International Journal of Computer Vision*, 4(3):185–210, 1990.

[14] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multiway cuts. In *ACM Symposium on Theory of Computing*, pages 241–251, 1992.

[15] H. Derin and H. Elliott. Modeling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Transactions on Pattern Analsis and Machine Intelligence*, 9(1):39–55, 1987.

[16] P. Ferrari, A. Frigessi, and P. de Sa. Fast approximate maximum a posteriori restoration of multicolour images. *Journal of the Royal Statistical Society, Series B*, 57(3):485–500, 1995.

[17] P. Ferrari, m. Gubitoso, and E. Neves. Reconstruction of gray-scale images. *Available from* `http://www.ime.usp.br/~pablo`, 1997.

[18] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[19] D. Geiger and A. Yuille. A common framework for image segmentation. *International Journal of Computer Vision*, 6(3):227–243, 1991.

[20] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[21] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.

[22] T. Hofmann, J. Puzicha, and J.M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analsis and Machine Intelligence*, 20(8):803–818, 1998.

[23] B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[24] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *European Conference on Computer Vision*, pages 232–247, 1998.

[25] H. Ishikawa and D. Geiger. Segmentation by grouping junctions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 125–131, 1998.

[26] G. Kant and X. He. Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems. *Theoretical Computer Science*, 172:175–193, 1997.

[27] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active countour models. *International Journal of Computer Vision*, 1(4):321–331, 1987.

[28] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[29] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. In IEEE Symposium on Foundations of Computer Science, 1999.

[30] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[31] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *Science*, 194:209–236, 1976.

[32] R.H.J.M. Otten and L.P.P.P van Ginneken. *The Annealing Algorithm*. Kluwer Academic Publishers, 1989.

[33] G. Parisi. *Statistical field theory*. Addison-Wesley, Reading MA, 1988.

[34] T. Poggio, E. Gamble, and J. Little. Parallel integration of vision modules. *Science*, 242:436–440, 1988.

[35] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[36] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on systems, Man, and Cybernetics*, 6(6):420–433, 1976.

[37] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *6th International Conference on Computer Vision*, 1998.

[38] R. Szeliski. *Bayesian modeling of uncertainty in low-level vision*. Kluwer Academic Publishers, 1989.

[39] R.S. Szeliski. Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–302, 1990.

[40] D. Terzopoulos. Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:129–139, 1986.

[41] V. Torre and T. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):147–163, 1986.

[42] E. Wasserstrom. Numerical solutions by the continuation method. *SIAM Review*, 15:89–119, 1973.

[43] G. Winkler. *Image analysis, Random Fields and Dynamic Monte Carlo Methods*. Springer Verlag, 1991.