

MULTI-RESOLUTION GRAPH CUTS FOR STEREO-MOTION
ESTIMATION

by

Joshua A. Worby

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of The Edward S. Rogers Sr. Department of
Electrical and Computer Engineering
University of Toronto

Copyright © 2007 by Joshua A. Worby

Abstract

Multi-Resolution Graph Cuts for Stereo-Motion Estimation

Joshua A. Worby

Master of Applied Science

Graduate Department of The Edward S. Rogers Sr. Department of Electrical and

Computer Engineering

University of Toronto

2007

This thesis presents the design and implementation of a multi-resolution graph cuts (MRGC) for stereo-motion framework that produces dense disparity maps. Both stereo and motion are estimated simultaneously under the original graph cuts framework [8]. Our framework extends the problem from one to five dimensions, creating a large increase in complexity. Using three different multi-resolution graph cut algorithms, LDNR, EL and SAC, we reduce the number of pixels m and the number of labels n that limit the $\alpha - \beta$ swap algorithm (with complexity $O(mn^2)$) required from the definition of our semi-metric smoothness function. This results in a reduction of computation time and the ability to handle larger images and larger label sets. The choice of the three MRGC algorithms to use in computation determines the appropriate level of accuracy and computation time desired.

Dedication

To my family, especially my grandparents, who always stress the value of education.

Acknowledgements

I am sincerely grateful to my supervisor, Professor W. James MacLean, for his invaluable guidance and support that was vital to the success of this thesis. His patience and understanding created the best environment for my personal success and learning.

I would like to express my thanks to my colleagues and friends in the Vision and Image Dynamics Lab and at the university, who were a source of support and encouragement throughout the project. They provided for some very useful discussions both on matters related to the project and to life outside the university. Without them, I most likely would not have my sanity, my health or my knowledge of what the city of Toronto has to offer.

The University of Toronto and MDA are also acknowledged for their financial and equipment support. In particular, I would like to acknowledge my supervisor at MDA, Piotr Jasiobedzki, without whom the opportunity for an internship, with interesting computer vision problems and access to a calibrated stereo system, would not have been possible.

Above all, I am deeply grateful to my family and friends who have made this journey possible for me. Their support throughout the whole two and half year journey, always ensured that I was on track.

Contents

Abstract	ii
Acknowledgements	iv
List of Tables	viii
List of Figures	xi
1 Introduction	1
2 Background	4
2.1 Image Formation: The Pinhole Camera	4
2.2 Matrix Projection Model	6
2.3 Motion Field and Optical Flow	7
2.3.1 Optical Flow Component Velocities	9
2.3.2 The Aperture Problem	10
2.3.3 Parametric Models for Optical Flow Computation	11
2.3.4 Non-Parametric Optical Flow Computation	12
2.4 Stereo	13
2.4.1 Epipolar Geometry	15
2.4.2 Stereo Rectification	16
2.4.3 Stereo Matching	18
2.5 Graph-Based Energy Minimization	19
2.5.1 A Bayesian Framework for Vision Problems	19

2.5.2	The General Form of Energy Functions	21
2.5.3	Multiway Cut Formulation	23
2.5.4	Multiway Cut Minimization	26
2.6	Maximum Flow	28
2.7	Literature Review	30
2.7.1	Stereo and Motion	30
2.7.2	Combining Stereo and Motion	32
2.8	Summary	33
3	Multi-Resolution Graph Cuts (MRGC) for Stereo-Motion	35
3.1	Stereo Vision Using Graph Cuts	36
3.1.1	The Data Term	36
3.1.2	The Smoothness Term	38
3.2	Motion Estimation Using Graph Cuts	39
3.2.1	The Data Term	40
3.2.2	The Smoothness Term	41
3.3	Extending Graph Cuts to Use Combined Stereo and Motion Constraints	42
3.3.1	The Data Term	44
3.3.2	The Smoothness Term	44
3.3.3	The Complete Energy Function	45
3.4	System Overview	45
3.5	Image Pyramid Stage	47
3.6	Move Space Stage	47
3.6.1	Graph Building	48
3.6.2	Minimizing the Energy Function	50
3.7	Upsampling and Disparity Propagation Stage	52
3.8	Various Multi-Resolution Graph Cuts (MRGC) Algorithms For Stereo- Motion	53

3.8.1	Label Disparity Neighbourhood Restricted MRGC (LDNR)	55
3.8.2	Expanding Label Disparity Neighbourhood at Every Iteration MRGC (EL)	59
3.8.3	Swap All Combinations LDNR-MRGC (SAC)	60
3.9	Summary	62
4	Results	64
4.1	Gaussian Pyramids	66
4.2	Number of Labels	67
4.3	Accuracy	74
4.3.1	Disparity Results	75
4.3.2	Analysis	84
4.4	Summary	99
5	Conclusions and Future Work	100
5.1	Future Work	102
	References	104

List of Tables

3.1	Alpha-beta swap edge weights	50
4.1	System input parameters	66
4.2	Labels versus time for all graph cuts algorithms	72
4.3	Accuracy versus time for stereo disparity maps	85
4.4	Accuracy versus time for motion disparity maps	86
4.5	Energy values for stereo-motion algorithms	97

List of Figures

2.1	Pinhole camera	5
2.2	World coordinate system to camera coordinate system	6
2.3	Optical flow component velocities	10
2.4	The aperture problem	11
2.5	Triangulation	14
2.6	Simple stereo system	15
2.7	Epipolar geometry	16
2.8	Stereo Rectification	17
2.9	4-point neighbourhood system	23
2.10	Smoothness priors	24
2.11	Multiway cut graph	25
2.12	Move space algorithm	27
2.13	A simple directed weighted graph	28
3.1	Example of image sampling	37
3.2	Stereo data image sampling insensitivity	38
3.3	Motion data image sampling insensitivity	40
3.4	Four image representation	43
3.5	System overview diagram	46
3.6	Image pyramid	47
3.7	Structure of a 1D two-terminal graph	48

3.8	Search trees	51
3.9	Improved augmenting path algorithm	52
3.10	Upsampling and disparity propagation	53
3.11	Label disparity neighbourhood	56
3.12	Label error propagation between pyramid levels	59
4.1	Tsukuba sequence	65
4.2	Tsukuba ground truth maps	67
4.3	Image pyramids for stereo or motion	68
4.4	Image pyramids for combined stereo-motion algorithms	69
4.5	Labels versus time for stereo algorithms	70
4.6	Labels versus time for motion algorithms	71
4.7	Labels versus time for all graph cut algorithms	73
4.8	Total execution time graph for a sample combined stereo-motion algorithm	73
4.9	Disparity map for the normal stereo graph cuts algorithm	76
4.10	Disparity map for the normal motion graph cuts algorithm	76
4.11	Disparity maps for multi-resolution graph cuts for motion	77
4.12	Level disparity maps for LDNR stereo-motion graph cuts	78
4.13	Disparity maps for LDNR stereo-motion graph cuts	79
4.14	Level disparity maps for EL stereo-motion graph cuts	80
4.15	Disparity maps for EL stereo-motion graph cuts	81
4.16	Level disparity maps for SAC stereo-motion graph cuts	82
4.17	Disparity maps for SAC stereo-motion graph cuts	83
4.18	Accuracy versus time for all graph cut stereo algorithms	85
4.19	Accuracy versus time for all graph cut motion algorithms	86
4.20	Distribution of pixel error and inaccurate labels for stereo graph cuts . .	88
4.21	Distribution of pixel error and inaccurate labels for motion graph cuts . .	89
4.22	Distribution of pixel error and inaccurate labels for LS motion graph cuts	89

4.23	Distribution of pixel error and inaccurate labels for LDNR motion graph cuts	90
4.24	Distribution of pixel error and inaccurate labels for EL motion graph cuts	90
4.25	Distribution of pixel error for LDNR stereo-motion graph cuts	91
4.26	Inaccurate label map for LDNR stereo-motion graph cuts	92
4.27	Distribution of pixel error for EL stereo-motion graph cuts	93
4.28	Inaccurate label map for EL stereo-motion graph cuts	94
4.29	Distribution of pixel error for SAC stereo-motion graph cuts	95
4.30	Inaccurate label map for SAC stereo-motion graph cuts	96
4.31	Total energy graph for stereo-motion SAC graph cuts algorithm	98

Chapter 1

Introduction

The goal of computer vision is to design an artificial system that models the world through the analysis of information from images. In general, researchers have had success interpreting this information via statistical or geometric techniques to produce complex computer vision systems. Nowadays, systems exist that are able to perform difficult tasks such as object recognition, object tracking, scene reconstruction, image restoration, navigation, robotics and medical applications.

In order to perform these tasks, a focus in computer vision has been placed on developing algorithms to handle *stereo-vision* and *motion estimation*. Stereo algorithms infer depth of perceived scene points from two images taken from different viewpoints. Motion algorithms determine scene structure from two temporally separated images. A key issue in both cases is the problem of *correspondence*: determining matching elements (scene points, features, lines) between images. The position difference between corresponding features is defined as the *disparity*. Disparity estimates for every single pixel in an image form a *dense disparity map*.

In general, when determining visual correspondence, we assume that the scene contains Lambertian surfaces without specularities, reflective surfaces or transparent objects. These assumptions allow for intensity based frameworks, where it is assumed that corre-

sponding pixels' intensity is constant. However, many problems arise in this framework. When images are taken from two different viewpoints or at two different instances in time, lighting conditions may vary or camera sensors may induce noise into the system. Scene points captured across multiple pixels cause problems with *image blurring*. *Textureless regions* do not provide enough information, requiring information to be propagated from surrounding pixels through spatial smoothness constraints. However, *depth discontinuities* break these smoothness constraints. *Occlusions* cause foreground objects to hide different parts of the background between the two images, causing pixels in one image to have no corresponding pixels in the other. With all these problems, determining a dense disparity map becomes difficult, with many candidate solutions possible.

To evaluate the large number of candidate solutions, we can use an optimization approach to identify the best solution. There are two steps involved when using the optimization approach. The first step is to define an objective function that encapsulates the constraints of an acceptable solution and provides a measure of goodness for a solution. Smaller values of the objective function relate to better solutions. Thus, we attempt to find the the *global minimum*, or the optimal solution, of the objective function. The second step is to develop a mechanism to determine the optimal solution by minimizing the objective function. Often, a compromise is required between steps one and two to simplify the optimization task. This compromise sometimes makes it difficult to reach the optimal solution.

The optimization approach provides a Bayesian framework for many vision problems. We are able to formalize constraints of the problem and global properties of the system by encoding them in the objective function. However, the optimization approach is computationally expensive. Minimization of the objective function is generally a NP-hard problem, making it impossible to find the global minimum.

Graph cuts [6, 7, 8, 5, 18, 28] is an example of an optimization approach that computes visual correspondence. The method incorporates a global energy function and a

maximum flow technique to provide very accurate results. However, like most optimization approaches, it has a high computational cost. The technique depends greatly on the number of pixels in the image, or image size, and the size of the disparity range, which translates directly to the number of labels. Nowadays, image sequences are generally 640×480 pixels in size or greater and typically contain scenes with large object or camera motion, making this method intractable and very slow.

In this work, we address the limitations of the graph cuts technique described above. The goal of this work is to provide a framework to compute visual correspondence, or dense disparity maps, with increased speed, reduced computational cost and the ability to handle larger images and disparities, all the while maintaining high accuracy. The key to achieving our goals is the implementation of a multiscale technique for graph cuts that encodes both the combined stereo and motion constraints. The basic assumption is that a multiscale approach allows for a method to quickly initialize the objective function closer to the global minimum than if it were left to its own devices. This speeds up the minimization step and allows for larger disparities and larger images. To improve the accuracy, imposing both stereo and motion constraints should result in greater accuracy than using only stereo or motion constraints separately.

This thesis is organized as follows: Chapter 2 describes the relevant background information necessary for a combined stereo-motion approach. The chapter begins with a discussion of the image formation process, motion analysis and stereo-vision. We next explain the graph cuts technique and the method in [5] to minimize the objective function. Chapter 3 provides a detailed description of the design of an objective function that encodes both stereo and motion constraints and the methodology involved in creating a multiscale method. We present the resulting disparity maps and their analysis in Chapter 4, with Chapter 5 providing conclusions and possible directions for future work.

Chapter 2

Background

In Chapter 1 we established the focus of this work; to provide a framework to compute dense disparity maps, with increased speed, reduced computational cost and the ability to handle larger images and disparities, with improved accuracy when compared to the traditional graph cuts approach. In order to achieve this, this chapter introduces some of the concepts applied throughout the thesis. We begin with a brief description of the image formation process and the matrix projection model in Section 2.1 and Section 2.2, respectively. Section 2.3 discusses the motion field and optical flow, which is then followed by a description of stereo-vision in Section 2.4. Next, we present the graph cuts technique for stereo computation of dense disparity maps, with focus on the graph creation and objective function formulation in Section 2.5 and a possible energy minimization technique used in Section 2.6. Finally, Section 2.7 provides a literature review of stereo algorithms and motion algorithms, with emphasis on methods that combine the two.

2.1 Image Formation: The Pinhole Camera

Image formation is the process in which a three dimensional scene is projected onto a two-dimensional surface. In the case of a typical camera aimed at a real-world scene,

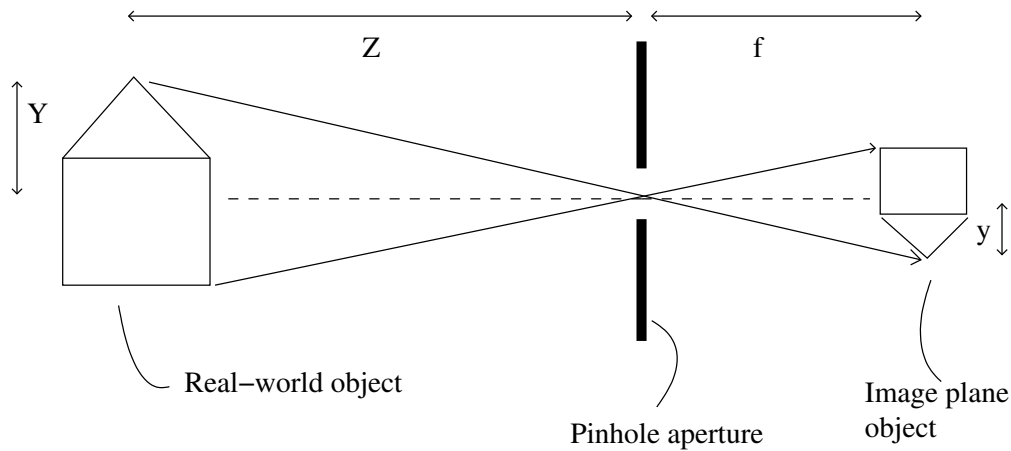


Figure 2.1: Example of a pinhole camera with focal length f , illustrating the projection of a real-world object of height Y and at depth Z from the aperture. This results in an object height of y in the image plane.

the two-dimensional surface coincides with a light-sensitive sensor. This sensor records light reflected from each point in the target scene to produce a corresponding “image” that can be reproduced at a later time. This description of the image formation process indicates a need for a one-to-one correspondence between visible scene points and image points, so that each image point records only the light emitted by its corresponding scene point. Developing a model of this geometric projection will aid our understanding of the image formation process.

The most common geometric model for this image formation process is the pinhole camera, illustrated in Figure 2.1. Light in the scene passes through a pinhole sized aperture onto the image plane where light in the real-world scene is recorded. Typically, we assume the camera origin to coincide with the camera’s aperture. We can formalize the resulting scene geometry by the following image point to scene point relationships:

$$x = f \frac{X}{Z} \quad (2.1)$$

$$y = f \frac{Y}{Z} \quad (2.2)$$

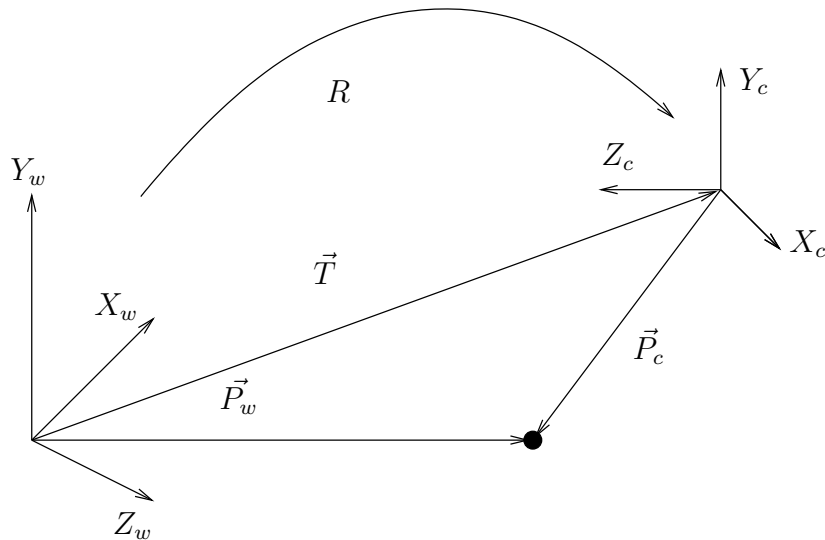


Figure 2.2: Transforming from the world coordinate system to the camera coordinate system requires a rotation R and a translation T .

2.2 Matrix Projection Model

To further formalize the geometric model presented in the previous section, we characterize the parameters of the underlying camera models. We begin by assuming some knowledge of the camera's characteristics, known as the camera's extrinsic and intrinsic parameters. These parameters define the transformations from a world reference frame to an image's pixel coordinate system.

The extrinsic parameters define the transformation from the world coordinate system to the camera coordinate system, which involve a rotation, R , and a translation, \vec{T} , illustrated in Figure 2.2. The translation describes the offset between the origin of the two systems, while the rotation brings into alignment the axes of the two systems. We represent the external matrix, M_{ext} , as

$$M_{ext} = [R, -R\vec{T}] \quad . \quad (2.3)$$

Similar to the extrinsic parameters, the intrinsic parameters define a transformation between two coordinate systems, the difference being a transformation from the camera coordinate system to the pixel coordinate system. These parameters formally characterize

the optical, geometric and digital characteristics of the viewing camera. In a pinhole camera model, we need to specify the focal length, f , the location where the optical axis intersects the image plane, known as the image centre, (o_x, o_y) , the pixel width and height scale factors, S_x and S_y respectively, and the skew introduced by the alignment of the camera optics, γ . We represent the intrinsic transformation matrix as

$$M_{int} = \begin{bmatrix} \frac{f}{S_x} & \gamma & o_x \\ 0 & \frac{f}{S_y} & o_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

resulting in the following linear matrix equation describing perspective projections

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = M_{int} M_{ext} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} \quad (2.5)$$

with image coordinates specified by

$$x_{im} = \frac{x_1}{x_3} \quad (2.6)$$

$$y_{im} = \frac{x_2}{x_3} \quad (2.7)$$

2.3 Motion Field and Optical Flow

Motion analysis studies the changes in the spatial and temporal domains discovered in an image sequence. Difficulties arise when attempting to determine which elements of one frame correspond to which elements of the next frame in a sequence [26], often known as the *correspondence problem*. The second major problem posed in motion analysis is the problem of reconstructing the 3D motion and structure of the scene given corresponding elements.

To tackle these problems, we introduce the notion of an idealized representation of visual motion termed the *motion field*. The motion field, often referred to as 2D motion,

is the resulting projection of the 3D velocity field onto the two dimensional image plane. Given a known focal length f , image points \vec{p} , and scene points \vec{P} , we represent the motion field equation as

$$\begin{aligned} \vec{u} = \frac{d\vec{p}}{dt} &= \frac{f}{P_3} \begin{bmatrix} 1 & 0 & \frac{-p_1}{f} \\ 0 & 1 & \frac{-p_2}{f} \\ 0 & 0 & 0 \end{bmatrix} \{ \vec{T} + \vec{\Omega} \times \vec{P} \} \\ &= \begin{bmatrix} 1 & 0 & \frac{-p_1}{f} \\ 0 & 1 & \frac{-p_2}{f} \\ 0 & 0 & 0 \end{bmatrix} \left\{ \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} + \begin{bmatrix} \Omega_2 f - \Omega_3 p_2 \\ -\Omega_1 f + \Omega_3 p_1 \\ \Omega_1 p_2 - \Omega_2 p_1 \end{bmatrix} \right\} \end{aligned} \quad (2.8)$$

where \vec{T} is the instantaneous translation velocity and $\vec{\Omega}$ is the rotational velocity.

Many instances occur where the motion field is not recoverable from an image sequence alone. For example, picture a mirror-textured, rotating sphere in a stationary environment. From the point of view of the image sequence, there is no change in terms of intensity values on the surface of the sphere, which reflects the scene points even though the sphere continues to rotate.

Rather than projecting from the scene to the image plane, we can rely on the information provided by the image sequence alone, such as image intensity values, to recover the motion field. The idea is to use image motion to approximate a recovery of the motion field, producing a 2D motion field termed the optical flow field. We estimate the *optical flow field* under the assumption that a scene's point intensity value will remain constant while in motion. This assumption is named the *Brightness Constancy Constraint* (BCC) [9, 16], and states that the intensity values, $I(x(t), y(t))$, of a particular scene point, $(x(t), y(t))$, do not change over short time intervals, Δt . Formally, the BCC is expressed by

$$\frac{dI(x, y, t)}{dt} = 0 \quad . \quad (2.9)$$

Using the chain rule for differentiation, we express this relation with respect to spatial and temporal partial derivatives:

$$\frac{dI(x, y, t)}{dt} = \frac{\delta I}{\delta x} \frac{dx}{dt} + \frac{\delta I}{\delta y} \frac{dy}{dt} + \frac{\delta I}{\delta t} = 0 \quad . \quad (2.10)$$

To simplify the notation, we use

$$u_x = \frac{dx}{dt} \quad , \quad u_y = \frac{dy}{dt} \quad (2.11)$$

$$I_x = \frac{\delta I}{\delta x} \quad , \quad I_y = \frac{\delta I}{\delta y} \quad , \quad I_t = \frac{\delta I}{\delta t} \quad (2.12)$$

resulting in

$$I_x u_x + I_y u_y + I_t = 0 \quad (2.13)$$

or in vector notation

$$\vec{\nabla} I_{\vec{x}}^T \vec{u} + I_t = 0 \quad , \quad \vec{\nabla} I_{\vec{x}} = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \quad (2.14)$$

or

$$\vec{\nabla} I^T \begin{bmatrix} \vec{u} \\ 1 \end{bmatrix} = 0 \quad \text{where} \quad \vec{\nabla} I = \begin{bmatrix} I_x \\ I_y \\ I_t \end{bmatrix} \quad \text{and} \quad \vec{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} . \quad (2.15)$$

We are now able to discern image motion given the spatiotemporal gradients (I_x, I_y, I_t) of an image sequence. However, there are still many problems to be addressed in regards to the computation of optic flow. Optical flow is an ill-posed problem, due to the *Aperture Problem* described in Section 2.3.2, and does not always coincide with the motion field. Other difficulties relate to violations of the BCC equation due to scene lighting conditions and object surface characteristics (such as untextured image regions), temporal aliasing, occlusions and errors in gradient estimates caused by the aggregation of information over a finite area.

2.3.1 Optical Flow Component Velocities

We introduce optical flow in terms of its component velocities. Figure 2.3 illustrates optical flow in terms of its two component vectors, \vec{u}_{\parallel} and \vec{u}_{\perp} . The component of the

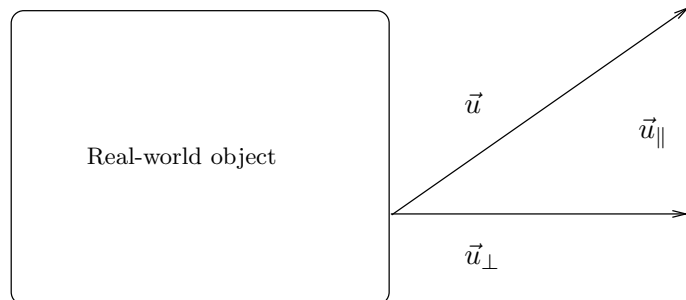


Figure 2.3: The motion of a real world object is represented by an optical flow vector, \vec{u} . This optical flow vector can be broken down into two orthogonal vectors, \vec{u}_{\parallel} and \vec{u}_{\perp} .

optical flow parallel to the object contour is described by \vec{u}_{\parallel} , while \vec{u}_{\perp} gives the component of the optical flow perpendicular to the object's contour.

2.3.2 The Aperture Problem

The main complication with the BCC equation is that the equation has only one constraint for the two unknowns of velocity, forming an ill-posed problem. We best illustrate this fact by isolating the measurable quantities in Equation 2.13 to arrive at

$$\vec{u}_{\perp} = -\frac{I_t}{\|\nabla I\|} \quad . \quad (2.16)$$

This problem is known as the *Aperture Problem* and can best be visualized in Figure 2.4. Given the viewable region of the circle and an edge of a moving square at time t (thick line) and at time $t + 1$ (thin line), we are only able to visualize a motion perpendicular to the image gradient, Figure 2.4(a), with no ability to retrieve the motion parallel to the image gradient, Figure 2.4(b).

To overcome the aperture problem, we estimate motion by grouping component velocities, requiring a minimum of two non-parallel component velocities. Therefore, motion estimation now becomes a matter of recovering these components accurately and then

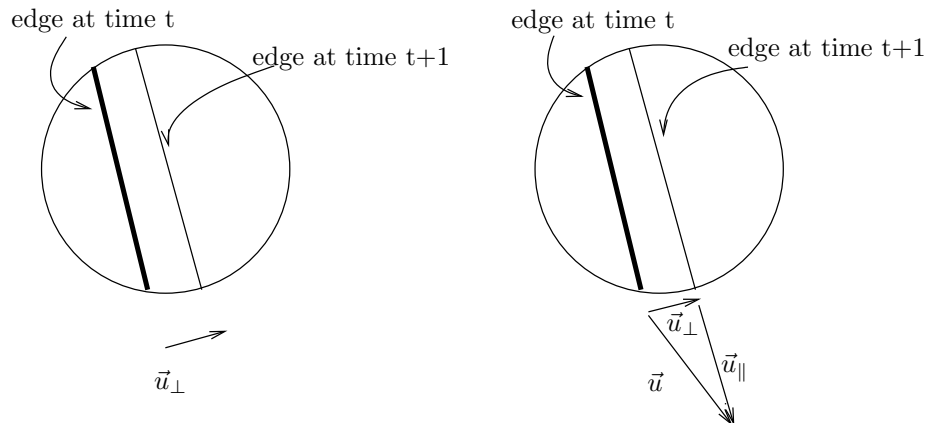


Figure 2.4: The aperture problem where (a) BCC only able to derive \vec{u}_\perp given a small viewing region while (b) shows the actual motion vector, \vec{u} of the moving square.

grouping then in an intelligent manner. Methods attempting to solve this problem are still areas of current research in computer vision.

2.3.3 Parametric Models for Optical Flow Computation

One method to compute optic flow vectors is to assume an underlying motion model for the image or some region. This way, we reduce the number of parameters needed to estimate the optic flow, making them less sensitive to noise. However, we still have the dilemma of deciding how large a region over which to group the optic flow vectors. If we use too large a region, we run the possibility of grouping optic flow vectors from two or more objects moving in different directions. For example, a region crossing an object boundary contains BCC's that are unrelated, thereby contaminating our motion estimates.

Typical parametric motion models are the constant motion model, the affine motion model, the projective motion (or planar) model and higher order parametric motion models. The constant motion model represents the optic flow vectors as simple translations over the entire region, where each location undergoes the same translation. The resulting

motion, \vec{u} , moves the image point \vec{x} to \vec{x}' via

$$\vec{x}' = \vec{x} + \vec{u} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} u_x \\ u_y \end{bmatrix} . \quad (2.17)$$

A second parametric model, the affine motion model, represents the possible motion an object can undertake as translation, shearing, scaling and rotation over the region.

We represent the affine motion model as

$$\vec{x}' = A\vec{x} + \vec{b} \quad \text{or} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} . \quad (2.18)$$

Higher order parametric models, such as the projective motion model, are outside the scope of this chapter and are left for the reader to explore at their own discretion.

2.3.4 Non-Parametric Optical Flow Computation

Originally, Horn and Schunck [16] estimated optic flow without an underlying motion model. Since the BCC equation 2.13 was an ill-posed problem, they imposed a second global constraint, known as the *smoothness constraint*, requiring neighbouring pixels to be spatially coherent.

Their approach utilized the following cost function:

$$C = \int \int_R (\alpha \zeta_c^2 + \zeta_b^2) dx dy \quad (2.19)$$

where α is a constant greater than 0. α determines the relative importance of the BCC term ζ_b , represented as

$$\zeta_b = I_x u_x + I_y u_y + I_t = 0 \quad (2.20)$$

and the ζ_c^2 term enforces the smoothness constraint via:

$$\zeta_c^2 = \frac{du_x^2}{dx} + \frac{du_x^2}{dy} + \frac{du_y^2}{dx} + \frac{du_y^2}{dy} . \quad (2.21)$$

Problems with this method arise in regions of the image where the optic flow may be discontinuous. Solving Equation 2.19 for a minimum value requires the calculus of variations, and may be non-trivial to solve for more general cost functions that can deal with discontinuities [24].

2.4 Stereo

In the field of visual perception, *stereopsis* is the process where we infer the depth, or distance of objects. Given multiple images of a scene taken from different viewpoints and known intrinsic and extrinsic parameters, we are able to discern the depth of an object through the process of *triangulation*. In the case of binocular stereo, we perform triangulation on the resulting images from two cameras in order to obtain the location of 3-D scene points. We illustrate this process in Figure 2.5 where we determine a scene point's location by the intersection of two rays. Each ray originates from the centre of projection of its respective camera, passes through its image point, and intersects with the ray from the other camera at the location of the scene point. In general, the two rays will not intersect in space so we estimate their point of intersection as the point of minimum distance from both rays in a least squares manner.

Stereo vision is similar to motion analysis in that it must solve the two main problems of correspondence and reconstruction. The difference between the two methods is in their respective setup. While motion has one camera creating two images separated temporally, stereo has two cameras, separated by some baseline distance, taking images at the same instance in time. Therefore, correspondence now becomes a search for matching points in the binocular image pair, often termed *stereo matching*.

Reconstruction, on the other hand, often deals with the interpretation of disparities of all the corresponding image points, where a disparity is defined as the difference between the location of a pixel in the left image and the location of its corresponding pixel in the

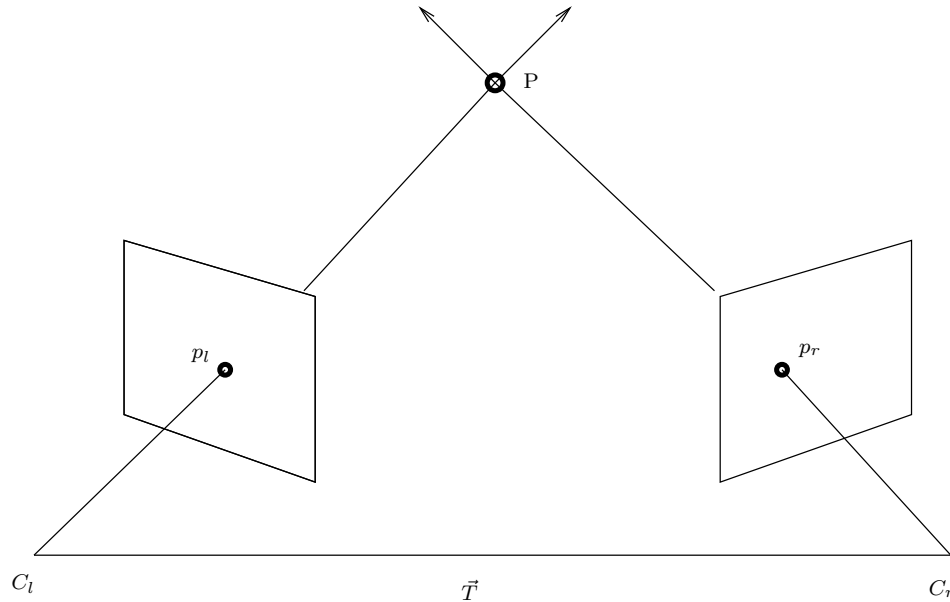


Figure 2.5: Triangulating a scene point's location from the intersection of two rays, originating from camera centres C_l and C_r .

right image. By grouping all the resulting disparities into a *disparity map*, we are able to create 3D map of the viewed scene. In the simplest case of a binocular setup, where the optical axes are parallel and perpendicular to the baseline, and the focal lengths f for both cameras are the same, we can use similar triangles to compute the distance to the scene point P . This is best illustrated in Figure 2.6 where we are given corresponding points in the left image x_l and in the right image x_r , the optical centres C_l and C_r for the left and right cameras respectively and the baseline T . Using similar triangles, we obtain

$$\frac{Z}{Z - f} = \frac{T}{T - x_r + x_l} = \frac{T}{T - d} \quad (2.22)$$

where disparity is $d = x_r - x_l$. The resulting distance Z is

$$Z = f \frac{T}{d} . \quad (2.23)$$

In this simple geometry, the disparity always lies along the scanline (a row with the same y value) of an image, thereby simplifying the correspondence search from a two-dimensional search (as in motion analysis) to a one-dimensional search.

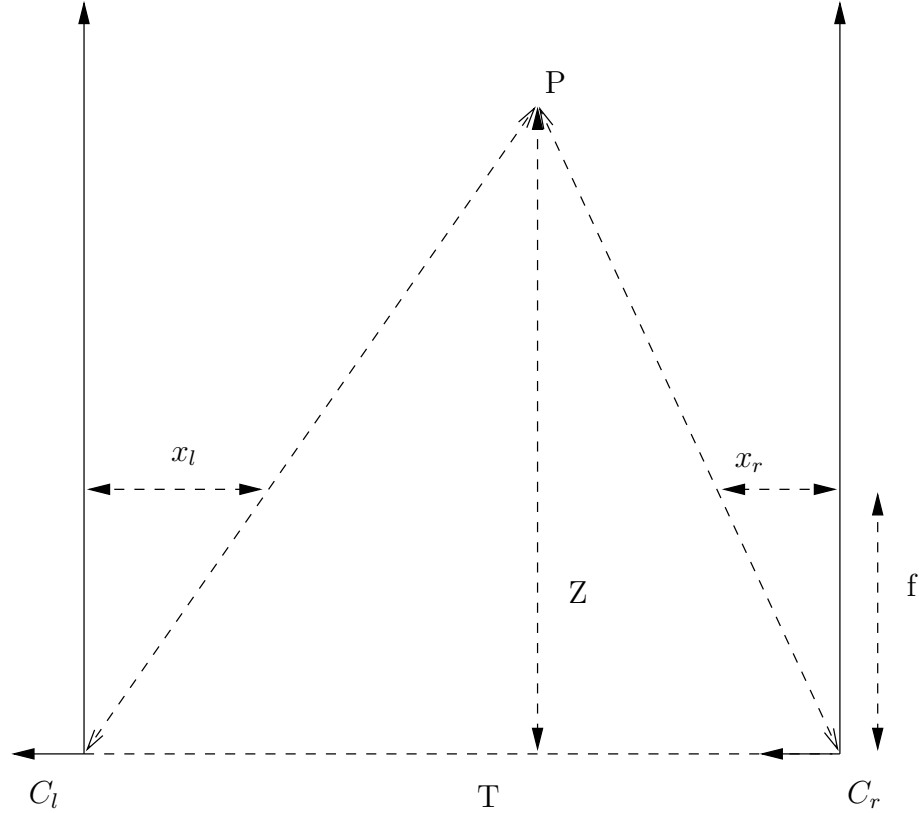


Figure 2.6: Given camera centres C_l and C_r , focal length f , baseline T and known point correspondences x_l and x_r , we are able to obtain distance Z to point P using similar triangles.

2.4.1 Epipolar Geometry

In the most general case of a stereo system, we get the system shown in Figure 2.7. The figure shows two pinhole cameras, their respective projection centres C_l and C_r and the baseline $\vec{T} = C_l - C_r$. The scene point P is referenced by two vectors $\vec{P}_l = [X_l, Y_l, Z_l]^T$ and $\vec{P}_r = [X_r, Y_r, Z_r]^T$, which pass through their respective image planes at image points $\vec{p}_l = [x_l, y_l, z_l]^T$ and $\vec{p}_r = [x_r, y_r, z_r]^T$. The relation between \vec{P}_l and \vec{P}_r is

$$\vec{P}_r = R(\vec{P}_l - \vec{T}) \quad (2.24)$$

and between a scene point and its projected points in the image planes is

$$\vec{p}_l = \frac{f_l}{Z_l} \vec{P}_l \quad \vec{p}_r = \frac{f_r}{Z_r} \vec{P}_r \quad . \quad (2.25)$$

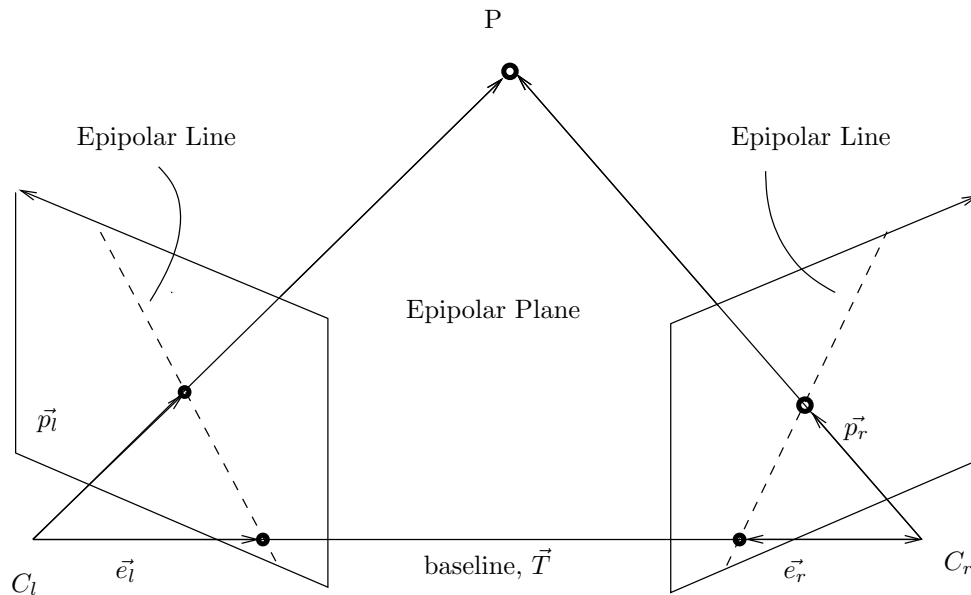


Figure 2.7: The epipolar geometry of the most general case of a stereo system.

The importance of epipolar geometry lies in the relationship between the *epipolar plane*, defined by C_l , C_r and P , and its lines of intersection with each image planes, called *epipolar lines*. The corresponding points p_r and p_l both lie on corresponding epipolar lines, thus rendering the search one-dimensional again. This is known as the *epipolar constraint*.

2.4.2 Stereo Rectification

To create the simple system of Figure 2.6 from the general system in Figure 2.7, we use a process known as *stereo rectification*. This process aligns the epipolar lines of the binocular image pairs, reducing the search for corresponding points down to the same row in the image.

The general idea is to rotate each camera around their optical centres, as shown in Figure 2.8. Each rotation is described as a non-singular 3×3 projection, or *homography*

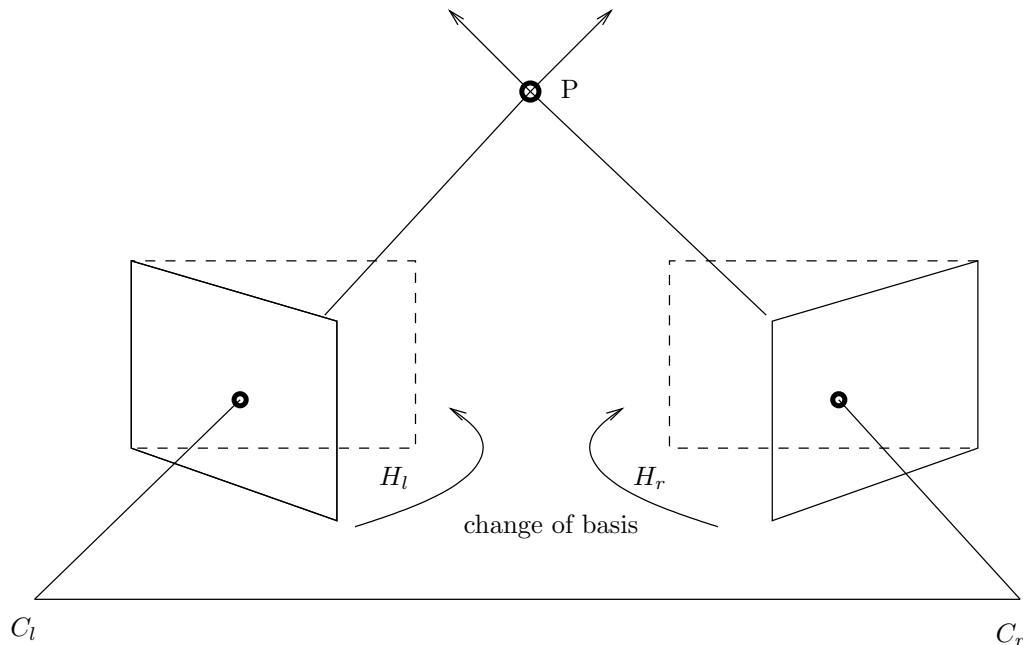


Figure 2.8: Stereo rectification.

matrix H , reprojecting pixels from an initial image to a transformed image according to

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} . \quad (2.26)$$

In this case, the initial stereo pair of images are rectified by applying two appropriate homographies, H_l and H_r . We compute H_l and H_r knowing the position and orientation of the two cameras given known camera intrinsic and extrinsic parameters. Applying the two homographies to the original images, we obtain the new image pair of rectified images. For each pixel (x', y') in the rectified image, its corresponding pixel (x, y) in the original image is computed using H^{-1} . This results in real-valued (non-integer) image coordinates, thus requiring a method to determine a pixel's intensity value, such as bilinear or biquadratic interpolation, from a neighbourhood of pixel intensity values.

2.4.3 Stereo Matching

In [23] stereo matching algorithms generally follow these four steps:

1. Matching cost computation: some common ones include squared intensity differences (SD), absolute intensity differences (AD), normalised cross-correlation, and binary matching costs based on features such as edges.
2. Cost (support) aggregation: the summing or averaging of the matching cost over a *support region* in the neighbourhood around the pixel.
3. Disparity computation / optimisation: selecting the best disparity estimate based on some cost function or error measure.
4. Disparity refinement: improving disparity estimates to sub-pixel accuracy.

The actual sequence of steps may vary and as such, depends on the specific algorithm used.

Stereo algorithms take two forms: (1) sparse methods; and (2) dense methods. Sparse methods reduce the search for correspondence to a distributed set of discrete features, such as edges or corners, producing sparse disparity maps. Dense methods, on the other hand, produce disparity values at every pixel. Typically they can be classed into *local* or *global* methods depending on the measure used in step 3 above. Local methods place more importance on the matching cost computation and cost aggregation steps, resulting in a greedy approach at each pixel. A classic example of a local method is performed by Lucas and Kanade [20] where a local constant motion model is determined by a weighted least squares solution. Global methods make explicit smoothness assumptions and then solve an optimization problem. This emphasizes the search for a disparity computation (step 3) that minimizes a global cost function, which combines data (step 1) and smoothness terms. This is best demonstrated in the original paper on optic flow by Horn and Schunck

[16]. Their global method uses a smoothness constraint and second order derivatives in an iterative process to determine optic flow.

2.5 Graph-Based Energy Minimization

Many computer vision problems can be viewed as labeling problems. To specify a labeling problem, we have a set of n sites, P , and a set of k labels, L , with the problem lying in the assignment of a specific label to every site. Thus a labeling is a mapping from P to L .

In this section, we discuss the steps involved in a specific type of graph-based energy minimization technique called graph cuts. The first step involves formulating a global energy function based on a Bayesian framework using Markov Random Fields (MRF's). MRF theory was first introduced to the vision community by Geman and Geman [13]. It provides a basis for modeling contextual constraints in visual processing and interpretation in the form of an energy function, thus enabling us to develop optimal vision algorithms when using optimisation principles. It is assumed that knowledge of MRFs is known by the reader and will not be covered in this thesis.

The use of MRFs to formulate an energy function lends itself nicely to a maximum a posteriori (MAP) estimation. Estimates are obtained by solving a multiway minimum cut problem on a graph. Methods that provide good approximations are two different iterative algorithms, called α -expansion and α - β swap. Finally, we discuss these algorithms and their design implications.

2.5.1 A Bayesian Framework for Vision Problems

Geman and Geman [13] introduced Markov Random Fields to the computer vision community. An MRF is composed of a set $P = \{1, 2, \dots, n\}$ of pixel sites p , a neighbourhood system $N = \{N_p | p \in P\}$ where each N_p is a subset of pixels in P describing the neigh-

bours of p , and a set of random variables $F = \{F_p | p \in P\}$. Each F_p is assigned a value in the label set L .

In [19], the Hammersley-Clifford theorem proves the equivalence between MRF's and Gibbs random fields. This theorem defines a clique as a set of sites where each member is a neighbour of all the other members. That way a Gibbs random field can be specified by the Gibbs distribution

$$P(f) = Z^{-1} \exp \left(- \sum_{c \in C} V_c(f) \right) \quad (2.27)$$

where C is the set of all cliques, Z is the normalizing constant and $V_c(f)$ are functions from a labeling to a real number, called clique potential functions. Assuming a four-point neighbourhood system N , we create clique potentials involving pairs of neighbouring pixels. We denote $f = \{f_p | p \in P\}$ as a particular configuration of the field F . Thus, probability of a particular configuration $P(f = F)$, abbreviated by $P(f)$, in the joint event $\{F_1 = f_1, \dots, F_m = f_m\}$, gives the joint distribution

$$P(f) = Z^{-1} \exp \left(- \sum_{p \in P} \sum_{q \in N_p} V_{p,q}(f_p, f_q) \right) . \quad (2.28)$$

In general, since the field F is not directly observable, we have to estimate its realized configuration based on the observation O . This is related to f by means of the likelihood function $P(O|f)$ via maximum a posteriori (MAP) estimation, which consists of maximizing the posterior probability $P(f|O)$. Bayes law tells us that the posterior probability can be represented as

$$P(f|O) = \frac{P(O|f)P(f)}{P(O)} \quad (2.29)$$

resulting in the following MAP estimate

$$P(f|O) \propto \arg \max_{f \in F} (P(O|f)P(f)) \quad (2.30)$$

All we require is a model for our likelihood function $P(O|f)$, usually given in our model for sensor noise. Given the observation O_p at pixel p , we assume that

$$P(O|f) \propto \prod_{p \in P} p(O_p|f_p) \quad (2.31)$$

holds when pixel noise is independent. If we further assume that

$$P(O_p|f) = C_p \cdot \exp(-D_p(f)) \quad \text{for } f \in L \quad , \quad (2.32)$$

where C_p is the normalizing constant and D_p is the data function that models the sensor noise of our system, then our likelihood function is

$$P(O|f) \propto \exp\left(-\sum_{p \in P} D_p(f_p)\right) \quad , \quad (2.33)$$

where $D_p(\cdot)$ is a data function that models the sensor noise of our system.

Our MAP estimate 2.30 now becomes

$$P(f|O) = \arg \max_{f \in F} \exp\left(-\sum_{p \in P} \sum_{q \in N_p} V_{p,q}(f_p, f_q) - \sum_{p \in P} D_p(f_p)\right) \quad (2.34)$$

resulting in the final form of the energy equation

$$E(f) = \sum_{p \in P} D_p(f_p) + \sum_{p \in P} \sum_{q \in N_p} V_{p,q}(f_p, f_q) \quad . \quad (2.35)$$

2.5.2 The General Form of Energy Functions

Equation 2.35 fits the form of an energy equation used in many vision applications

$$E(f) = E_{data}(f) + \lambda E_{prior}(f) \quad (2.36)$$

with λ controlling the importance of both energies. A smaller λ places more weight to the data energy term while a larger λ puts more emphasis on the prior energy term.

The most common constraint in vision applications is the data constraint, represented by the *data energy* term, $E_{data}(f)$. Typically, we assign a small cost to labelings close to the data and a large cost to labelings that do not agree with the data, thereby, fitting the noise model of our imaging system. As is apparent from 2.35 and 2.36, our data term is

$$E_{data}(f) = \sum_{p \in P} D_p(f_p) \quad . \quad (2.37)$$

A common data energy term in visual correspondence models the difference in pixel intensities between two corresponding pixels in both images. Given corresponding pixels p and p' in the primary and secondary images, respectively, let I_p and I'_p be their pixel intensities. Thus, a common data energy term used is $(I_p - I'_p)^2$ when determining the correctness of a correspondence match. However, as is evident from the data energy term, unless the pixel intensities are quite close in range, the cost of the data term increases rapidly the more the intensities values vary.

Why not use our preconceived ideas of an ideal solution, which we know contains some structured patterns, in constraining the results further rather than relying solely on the data energy? Our prior knowledge might make some labelings likely while others unlikely. It is this prior knowledge that is encoded in the *prior constraint*, represented by the *prior energy* term, $E_{prior}(f)$ in 2.35. We assign a heavy cost to the labelings f which do not fit our prior knowledge.

Designing the prior energy term is rendered more difficult in that it depends on the problem at hand. Most vision problems try to enforce some spatial constraint, such as a *smoothness constraint* [16], which ensures our estimate is smooth everywhere. However, in such systems, difficulties arise when there is an abrupt change, such as object boundaries, thus requiring different smoothness priors. Many different kinds of priors exist, but in this work we concentrate on smoothness priors. Therefore, we now refer to the prior term, E_{prior} , as the smoothness energy term, $E_{smooth}(f)$.

Smoothness terms generally model the interactions between a pixel p and its neighbourhood, N_p . For the purpose of this paper, it is assumed that we are using a four neighbourhood system illustrated in Figure 2.9 modeling clique pairs rather than quads. Given this neighbourhood system, the smoothness energy term takes the form

$$E_{prior}(f) = E_{smooth}(f) = \sum_{\{p,q\} \in N} V_{p,q}(f_p, f_q) \quad (2.38)$$

where $V_{p,q}(f_p, f_q)$ is called the neighbour interaction.

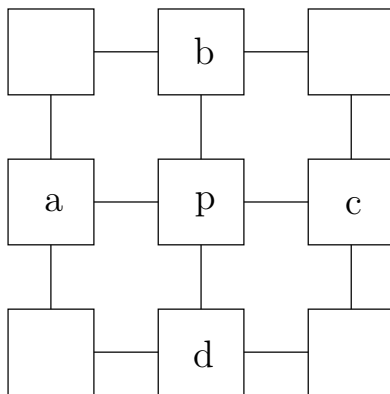


Figure 2.9: A 4-point neighbourhood system for pixel p . p 's neighbourhood consists of pixels a, b, c, d and is represented as $N_p = \{a, b, c, d\}$.

In [28], three types of smoothness priors are discussed: (1) everywhere smooth prior; (2) piecewise constant prior; and (3) piecewise smooth prior. The everywhere smooth prior (Figure 2.10a) assigns low cost to labelings which are smooth everywhere. Most image sequences tend to have discontinuities, creating troubles for this type of approach where results tend to be over-smoothed around discontinuities. The second prior, the piecewise constant prior (Figure 2.10b), assigns low cost to labelings which consist of one or several pieces with constant labels. However, this method is not expressive enough because real data can vary smoothly within each piece. This differs from the piecewise smooth prior (Figure 2.10c), which accounts for this deficiency. Penalties are allowed to grow up to a maximum penalty, thus ensuring penalty assignments never grow too large and allowing discontinuities to occur. Consequently, this approach lends itself to a wider range of problems.

2.5.3 Multiway Cut Formulation

The simplest smoothness prior that allows discontinuities is the piecewise constant prior.

This is modeled as

$$V_{p,q}(f_p, f_q) = u_{p,q} \cdot \delta(f_p \neq f_q) \quad (2.39)$$

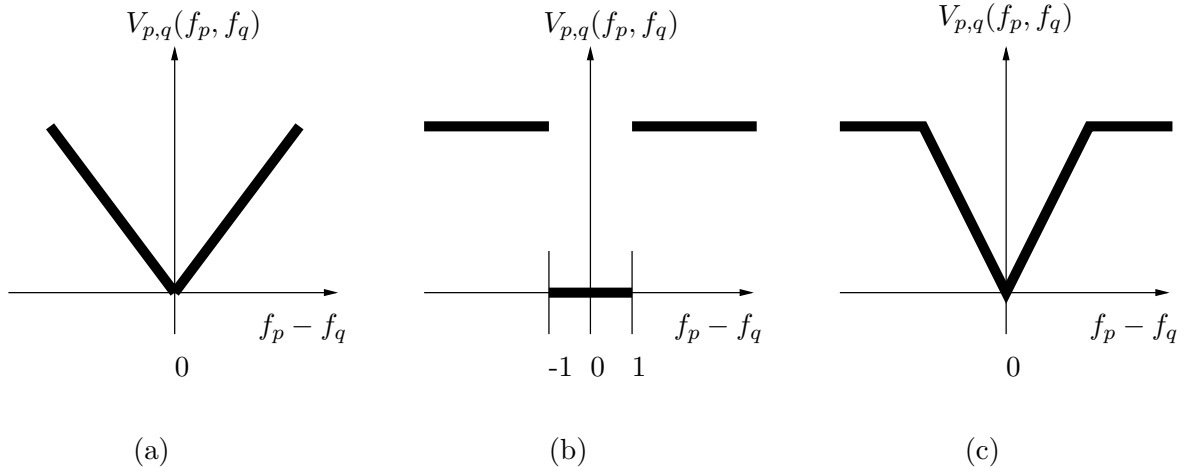


Figure 2.10: Graph of $V_{p,q}(f_p, f_q)$ for (a) everywhere smooth prior, (b) piecewise constant prior, and (c) piecewise smooth prior.

where

$$\delta(f_p \neq f_q) = \begin{cases} 1 & \text{if } f(p) \neq f(q), \\ 0 & \text{otherwise} \end{cases} \quad (2.40)$$

and $u_{p,q}$ is a constant. This results in the *Potts* model energy equation

$$E_P(f) = \sum_{p \in P} D_p(f_p) + \sum_{\{p,q\} \in N} u_{p,q} \cdot \delta(f_p \neq f_q) \quad . \quad (2.41)$$

According to [28, 7, 8], minimizing the Potts energy $E_P(f)$ can be solved by computing a minimum cost multiway cut on a certain graph. For example, consider the graph $G = \langle V, E \rangle$ in Figure 2.11(a) with non-negative edge weights and a set of terminals $L \subset V$. A *multiway cut* C occurs when a subset of edges $C \subset E$ completely separates the terminals in the induced graph $G(C) = \langle V, E - C \rangle$, as shown in Figure 2.11(b) where dotted lines indicate the cuts made. This multiway cut problem is a generalization of the two-terminal graph cut problem.

To solve this multiway cut problem, we begin by constructing the graph in Figure 2.11a. Vertices in the graph are the set $V = P \cup L$ where the terminals are the set of possible labels $L = \{l_1, l_2, \dots, l_k\}$, called l-vertices, while the pixels are elements of $P = \{1, \dots, p, q, \dots, n\}$, called p-vertices. There are two sets of edges, t-links and n-links.

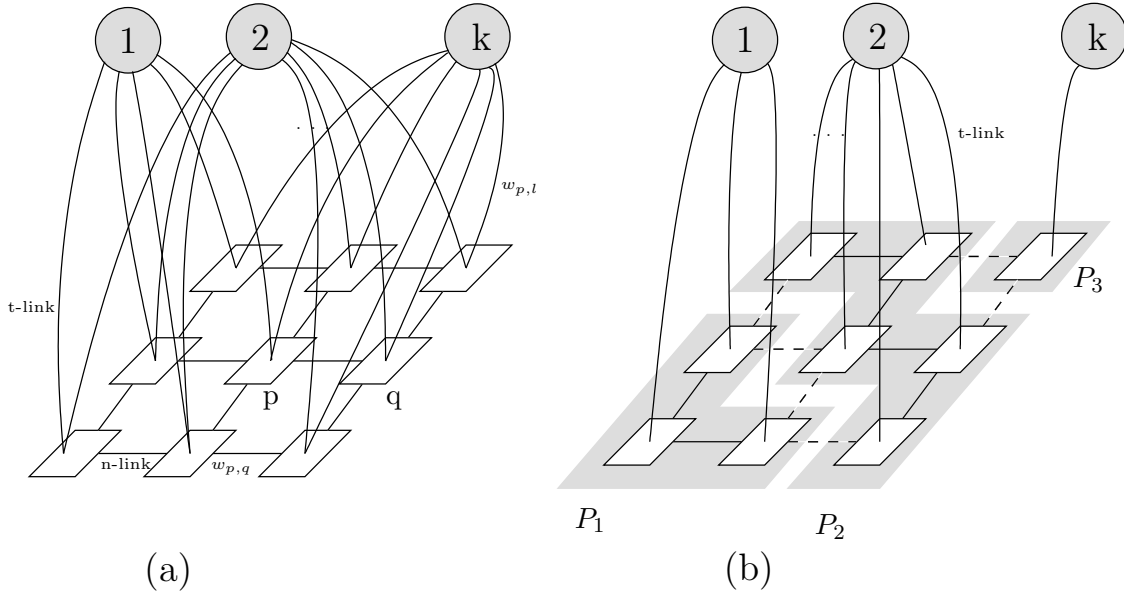


Figure 2.11: An example of (a) a graph $G = \langle V, E \rangle$ with (b) illustrating the induced graph by a multiway cut C where dotted lines indicate cut edges.

The latter is the set of edges connecting neighbouring pixels, E_N , while the former, E_T connects terminals to p-vertices. Edge weights for n-links and t-links, respectively, are

$$w_{p,q} = u_{p,q} \quad \text{and} \quad (2.42)$$

$$w_{p,l} = K_p - D_p(l), \quad (2.43)$$

where K_p is constant that is large enough to ensure positive weights (i.e. $K_p > \max D_p(l)$).

Figure 2.11b demonstrates an induced graph from a multiway cut. The keys to the graph are that each p-vertex is connected to at most one terminal, no two terminals are directly connected and there is no direct path between any two terminals. Thus, edges of the graph are $E = E_N \cup E_T$.

A multiway cut C corresponds to the labeling f^C which assigns the label l to all pixels p which are t-linked to the l -vertex in $G(C)$. Therefore, if C is a multiway cut on G , the cost of the cut is

$$|C| = E_P(f^C) \quad . \quad (2.44)$$

The proof lies in the fact that the cost of the cut $|C|$ is the sum of the weights of the n-links and t-links in C . Given that $f^C(p) \neq f^C(q)$, the sum of the n-links in C is

$$\sum_{\{p,q\} \in N} w_{p,q} \cdot \delta(f^C(p) \neq f^C(q)) \quad (2.45)$$

and the sum of the t-links is

$$\begin{aligned} \sum_{p \in P} \sum_{\substack{l \in L \\ l \neq f^C(p)}} w_{p,l} &= \sum_{p \in P} \sum_{\substack{l \in L \\ l \neq f^C(p)}} (K_p - D_p(l)) \\ &= \sum_{p \in P} \sum_{\substack{l \in L \\ l \neq f^C(p)}} K_p - \sum_{p \in P} \sum_{\substack{l \in L \\ l \neq f^C(p)}} D_p(l) \end{aligned} \quad (2.46)$$

This can be rewritten as

$$(|L| - 1) \sum_{p \in P} K_p - \sum_{p \in P} \sum_{l \in L} D_p(l) + \sum_{p \in P} D_p(f^C(p)) \quad . \quad (2.47)$$

Note that the first two terms are constants since only the last term depends on C . Therefore, we obtain $E_P(f^C)$ from 2.41 by adding the cost in 2.45 and 2.47.

2.5.4 Multiway Cut Minimization

Given the NP-hardness of the problem [7, 8, 28], an approximated solution is required. The algorithms presented in this paper generate a cut C such that f^C is a local minimum of the posterior energy function in 2.41 with respect to very large move spaces. A *move space* allows a large number of pixels to change their labels. This differs from the standard move spaces, exhibited in algorithms such as simulated annealing [13], where only one pixel's label changes. Consequently, the algorithm approaches convergence at a faster rate.

First, let us define a move space. Given a set of all labelings F , a *move* is a pair of labelings $(f, f') \in F \times F$. Therefore, a set of moves $M \subset F \times F$ is called a move space. If $(f, f') \in M$, then we say that f is within one move from f' . A labeling is a local minimum with respect to some move space M if $E(f) \leq E(f')$ for any $(f, f') \in M$.

1. Start with an arbitrary labeling f
2. Set success = 0
3. For each possible move space
 Find $\hat{f} = \arg \min E(f')$ among f' within one move space of f
 if $E(\hat{f}) < E(f)$, set $f = \hat{f}$ and success = 1
4. If success == 1, go to step 2
5. Return f

Figure 2.12: General outline of the iterative algorithms (α - β swap and α -expansion). Algorithm courtesy of [28].

We focus our attention on two types of move spaces for energy minimization: (1) the *alpha-beta swap* move space; and (2) the *alpha-expansion* move space. The swap move space is indexed by a pair of labels, $(\alpha, \beta) \subset L$ with the general idea being that that some pixels labeled α change their label to β while some pixels labeled β change their label to α . Meanwhile, the second move space, alpha-expansion, is indexed by a single label $\alpha \subset L$. Pixels in the graph are labeled α or $\bar{\alpha}$. The options in this move space are for pixels to retain their current label or to change their label to α .

Both iterative algorithms are quite similar in structure, as summarized in Figure 2.12. The algorithm starts with some arbitrary labeling f and continues searching for a labeling \hat{f} within one move from f that gives the lowest energy among all labelings f' within one move of f . The greedy nature of selecting the best solution at every *cycle* (steps 2-4) of the algorithm results in fast convergence.

The difference in the two algorithms lies in their approach to graph creation. The swap algorithm dynamically creates a separate two-terminal graph for every possible label pairing $(\alpha, \beta) \subset L$ for every *iteration* (step 3). Each graph is composed a differing number of nodes (pixels), labeled (α or β), connected to two terminals, the α terminal and the β terminal. The result is the creation of $|L|^2$ graphs. However, the expansion

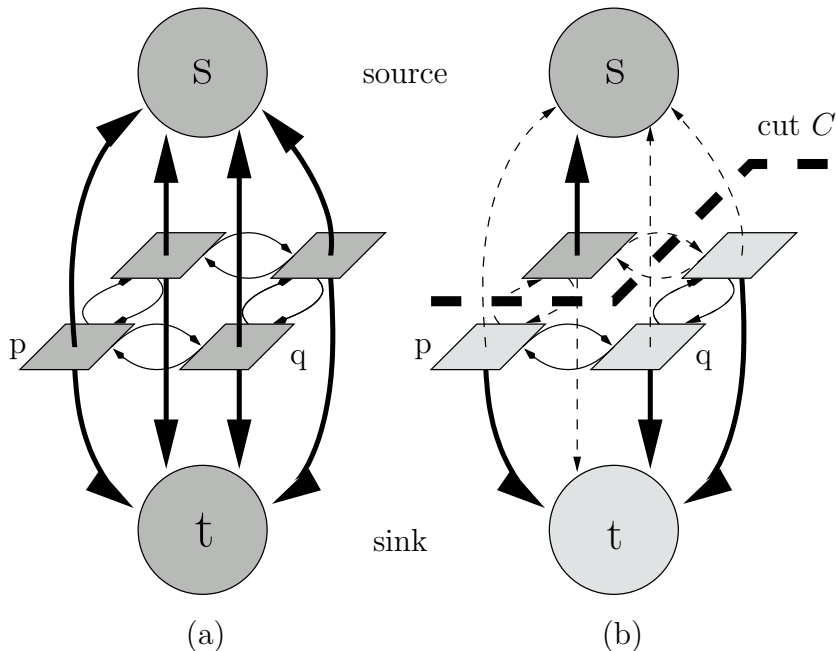


Figure 2.13: Example of (a) a simple binary directed weighted graph with (b) a cut C on the graph with cut edges denoted by dashed lines.

algorithm creates a two-terminal graph for every label $\alpha \in L$ at every iteration. The nodes are labeled α or $\bar{\alpha}$, resulting in $|L|$ graphs. For further details of the graph creation process, see Section 3.

2.6 Maximum Flow

In the previous section, we focused on the creation of directed weighted graphs from energy functionals of the form

$$E(L) = \sum_{p \in P} D_p(L_p) + \sum_{\{p,q\} \in N} V_{p,q}(L_p, L_q) \quad (2.48)$$

where $L = \{L_p | p \in P\}$ is a labeling of image P , $D_p(\cdot)$ is a data penalty function, $V_{p,q}$ is an interaction potential, and N is a set of all pairs of neighbouring pixels. We will concentrate on the case of graphs with two terminals, the source s and sink t , illustrated in the simple example of Figure 2.13.

The desired s/t cut C partitions the nodes in the graph into two disjoint subsets S , containing the source s , and T containing the sink t . The cost of the cut $C = \{S, T\}$ is the sum of the costs of the edges along the border between S and T with edge weights calculated according to the move space implemented, either α - β swap or α -expansion (discussed in further detail in Chapter 3). Finding a cut that has the minimum cost among all possible cuts is known as the *minimum cut* problem.

A solution to the minimum cut problem can be likened to finding a *maximum flow* from the source to the sink. The idea of maximum flow is analogous to finding the maximum amount of allowable water flow passing through a network of pipes, with pipe capacities equal to the edge weights, from the source to the sink. The theorem in [11] states that a maximum flow from s to t saturates a set of edges in the graph, dividing the nodes into two disjoint parts $\{S, T\}$ corresponding to a minimum cut, thereby creating a “duality” relationship between maximum flow and minimum cut. Therefore, the cut in Figure 2.13 corresponds to an assignment of pixels (nodes) to labels (terminals), creating a labeling with the energy minimized.

In general, maximum flow algorithms can be classed into two groups: (1) *push-relabel* methods and (2) *augmenting paths* algorithms [25]. Push-relabel algorithms work in a more localized manner than augmenting path algorithms by working on one vertex at a time, looking only at the neighbours in the residual network. The key to the algorithm is the height function of all vertices in the graph. Initially, the source’s height is equal to the number of vertices, while the sink is assigned a height of zero. Flow passes through the graph, from the source to the sink, in a downhill fashion, with excess flow gathering in each node’s reservoir. If a node can no longer pass excess flow to nodes downhill from it, the node is reassigned a height equal to its highest neighbouring node plus one. The process continues until undeliverable flow drains back to the source, meaning that the source and sink have been separated by saturated edges. The saturated edges signify the minimum cut necessary on the graph.

On the other hand, augmenting path algorithms work on the entire network for maximum flow and depend on three important ideas: residual networks, augmenting paths, and cuts. These algorithms are iterative and begin with no flow. At each iteration, the flow is increased by finding an augmenting path from the source to the sink along which we can send more flow. We then augment the flow along this path, saturating edges as we go. This process is repeated until no further augmenting path exists, yielding the maximum flow.

2.7 Literature Review

Stereo correspondence and motion analysis have long been central research problems in computer vision, producing a large expanse of research. With most of the work being placed on either method separately, it becomes difficult to gauge the progress of the field. Previous surveys on stereo vision [23] and optical flow [1] provide a good jumping-off point for an understanding of the algorithms already created, but little has been done to review recent research that incorporates both methods. It is the purpose of this section to provide an evaluation of these fused techniques.

Motivated by modern applications such as view synthesis and image-based rendering that require disparity estimates in all regions, including occluded and untextured regions, we focus our attention on dense methods for visual correspondence. We begin with a discussion of the key research in motion analysis and stereo vision, followed by a discussion of graph cuts. Finally, we survey previous techniques that have combined both stereo and motion.

2.7.1 Stereo and Motion

In sections 2.3 and 2.4, we showed that stereo and motion handle similar problems with motion viewed as a generalization of stereo. Motion performs a two-dimensional search

for correspondence while stereo searches in one dimension given rectified image pairs. Therefore, we treat them as one and the same problem and provide a review on stereo and motion methods as a whole.

There are two main classes of stereo and motion methods: (1) *local* methods; and (2) *global* methods. Local methods, often called window-based methods or correlation-based methods, find the optimal displacement of a fixed sized region between two consecutive frames. Typically, these methods compare intensities within the region according to some likeness measure, assume a certain motion model, and select the best choice in a greedy fashion. The main obstacles for local methods lie in the choice of window size and the assumption as to which motion model to incorporate over the region. Shiftable windows [4] and windows with adaptive sizes [21] are common techniques for varying the window size. In perhaps the most well known local method, Lucas and Kanade [20] use a local constant motion model for the optical flow and determine a weighted least squares solution.

Global methods provide greater accuracy compared to local methods, but have the disadvantage of creating a higher computation cost. Generally, they formulate a global energy function composed of a data term and a smoothness term.

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d). \quad (2.49)$$

Measuring the agreement between the disparity function d and the input image pair is encoded in the data term, $E_{data}(d)$. Meanwhile, the smoothness assumptions made by the algorithm are embedded in the smoothness term, $E_{smooth}(d)$, which is made more tractable by restricting the smoothness term to only neighbouring pixels. In regularisation techniques [16], Horn and Schunck are able to overcome the aperture problem but then encounter problems at object boundaries. Brox *et al.*[24] handle object boundaries by employing robust estimators, developed by Black and Anandan [3], on both the data and smoothness terms of their global energy function. This is the current state of the art in optical flow computation. Other global methods use differing methods to minimize the

global energy function. Some methods are simulated annealing [13], highest confidence first [10] and mean-field annealing [12].

In recent research, a promising method for global energy minimization has been developed, called graph cuts. Graph cuts, in cooperation with maximum flow techniques, provides a graph-based method for energy minimization [6, 7, 8, 5, 18, 28] that is more efficient than simulated annealing [13]. They handle discontinuities quite well, while achieving 98% accuracy on real data with ground truth.

2.7.2 Combining Stereo and Motion

For the past twenty years [30], research has been performed on incorporating both stereo and motion. This combined stereo-motion framework is based on four images of a stereo sequence, which are two stereo image pairs separated temporally. We refer to this type of framework as *combined stereo-motion*.

Most of the work to date focused on sparse image features. One approach to combined stereo-motion is to use an iterative method on a sparse set of features. Wang and Duncan [29] iteratively separated the dominant motion in the input set until they were left with a set of outliers, thereby separating the motions present in the stereo sequence into separate layers. They were able to attain sub-pixel accuracy of disparity estimates, absolute structure and rigid body motions without a scale factor ambiguity. Another approach is to use a probabilistic framework to disambiguate feature correspondences. Ho and Pong [15], used a probabilistic method with relaxation labeling to integrate the four images in the combined stereo-motion framework. They were able to obtain optical flow vectors for their set of sparse features with less sensitivity towards noise than previous methods.

In contrast, dense methods for combined stereo-motion provide a disparity estimate value for every pixel. Some approaches use an iterative method to segment the image [29] into regions of motion. By assuming a single dominant motion in a data set, they determine the dominant motion and view everything else as outliers. Another approach

is to use batch computation on the whole stereo image sequence [31] and [27] (scene flow). This approach benefits from the temporal knowledge of the whole sequence simplifying correspondence. However, both methods are heavily reliant on the computation of optic flow and thusly, they inherit all optic flow computation problems. A probabilistic framework [17] provides another approach for combined stereo-motion. By incorporating a single coherent probabilistic framework on four images, they are able to incorporate random variables for occlusions and discontinuities. The downside to their approach is the high computational cost involved.

The key problem with all combined stereo-motion methods is that they are more computationally expensive than computing stereo or motion separately. This is even more pronounced for the dense methods considering correspondence must be performed on every pixel for all four images.

2.8 Summary

This chapter introduces the concepts of motion and stereo in computer vision. Motion studies the changes in the spatial and temporal domains in an image sequence to recover a two-dimensional motion field called optic flow. We make the assumption that a pixel's intensity value remains constant while in motion, known as the Brightness Constancy Constraint (BCC). Problems arise due to the Aperture Problem, the optical flow not coinciding with the motion field, violations of the BCC equation, lighting conditions, object surface characteristics and occlusions.

Stereo vision uses scene and camera geometry to infer the depth to objects. This is done through the process of triangulation, given multiple images and known intrinsic and extrinsic parameters. Given rectified images, the correspondence problem becomes a one-dimensional search along the scanline. Finding corresponding points for every point

in the image, we obtain a dense disparity map, which helps infer distance from its inverse relationship with disparity.

Both stereo and motion have local and global methods. One successful global method is graph cuts, a graph-based energy minimisation technique based on a Bayesian framework. Its energy function is the sum of a data energy function and a smoothness energy function. The data energy function encodes the constraints of the data, while the smoothness energy function encodes the smoothness assumptions made by the algorithm. Minimising the energy function is equivalent to finding a multiway cut on the graph. To find this cut, maximum flows techniques are employed. In Chapter 3, we provide a detailed description of the design of a multi-resolution graph cuts algorithm for stereo-motion.

Chapter 3

Multi-Resolution Graph Cuts (MRGC) for Stereo-Motion

This chapter describes the implementation of a multi-resolution graph cuts (MRGC) method that uses combined stereo and motion constraints for visual correspondence. In Section 3.1, we begin with an analysis of existing graph cuts methods for stereo, identifying areas for improvement. This will allow for an explanation of the changes needed to the graph cuts framework to allow for motion estimation (Section 3.2).

Understanding the design decisions necessary to successfully implement graph cuts for stereo and motion separately, we present a novel framework to extend the original graph cuts technique to apply combined stereo and motion constraints (Section 3.3). We will provide a high-level overview of the system in Section 3.4, with a break down of the necessary stages in Sections 3.5, 3.6 and 3.7. Finally, we describe three possible algorithms under the graph cuts framework to compute visual correspondence using combined stereo and motion constraints (Section 3.8). They are (1) Label Neighbourhood Restricted Multi-Resolution Graph Cuts (LDNR), (2) Expanding Label Neighbourhood at Every Iteration Multi-Resolution Graph Cuts (EL) and (3) Swap All Combinations of LDNR (SAC).

3.1 Stereo Vision Using Graph Cuts

In this section, we analyze the original graph cuts technique for stereo [6, 7, 8, 5, 18, 28]. We begin with a description of the energy function used. This energy function is established in the general form of an energy equation used in computer vision as

$$E(f) = E_{data}(f) + \lambda E_{prior}(f) \quad (3.1)$$

where f is a labeling of the image. Next, we look at the design of the data term $E_{data}(f)$, noting that it is less sensitive to image sampling. Lastly, we examine the smoothness term $E_{smooth}(f)$ (weighted by λ), paying attention to its formulation as a metric or semi-metric and the effect that this has on the choice of graph building algorithm to use.

As we will see, the formulation of both the data term and smoothness are the two key factors for a successful implementation of the original graph cuts technique.

3.1.1 The Data Term

The data term E_{data} in the energy function evaluates the level of correspondence between values in the input image pair. In the original graph-cuts-stereo system, two key design decisions were made to improve the system:

1. Rectified stereo image sequences were assumed. This reduced the search for stereo correspondence from a two-dimensional search to a one-dimensional search along the horizontal scanlines of an image, increasing the speed of the system.
2. The system was made less sensitive to image sampling effects of camera sensors. This improved the accuracy of the system but had a negative effect of increasing the computational expense of the system.

Making the system insensitive to image sampling [2, 28] affected the formulation of the data term. Depending on the discretisation process of the differing cameras, pixel

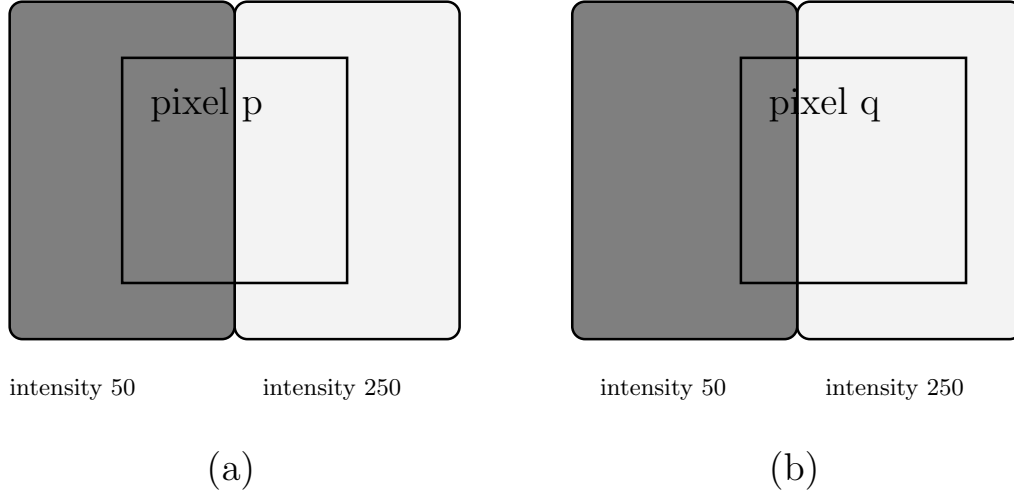


Figure 3.1: Example of the discretisation of an image in (a) the left image and (b) the right image. Corresponding pixels p and q get differing intensities even if there is no camera noise.

intensities could vary greatly; for example, if a pixel were located along the border of a scene patch with a high intensity gradient. Figure 3.1 shows this sampling effect if a point p (in Figure 3.1a) corresponds to point q (in Figure 3.1b). There are two regions with intensity values of 50 and 250. Pixel p overlaps the regions equally giving a pixel intensity value of 150. Meanwhile, pixel q composes a quarter of the surface patch with intensity value 50 with the rest lying in the region with intensity value 250 resulting in a pixel intensity value of 200. The corresponding pixels' intensity values differ by 50; a value too large to be accountable due to camera noise.

Since the search is one-dimensional, measurements of how well p fits into the real valued range of disparities $(d - \frac{1}{2}, d + \frac{1}{2})$ are found, as illustrated in Figure 3.2.

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

$p + x$ stands for a pixel which has coordinates of p shifted by disparity x . I' represents intensities in the right image. Fractional values I'_{p+x} are obtained by linear interpolation between discrete pixel values. For symmetry,

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

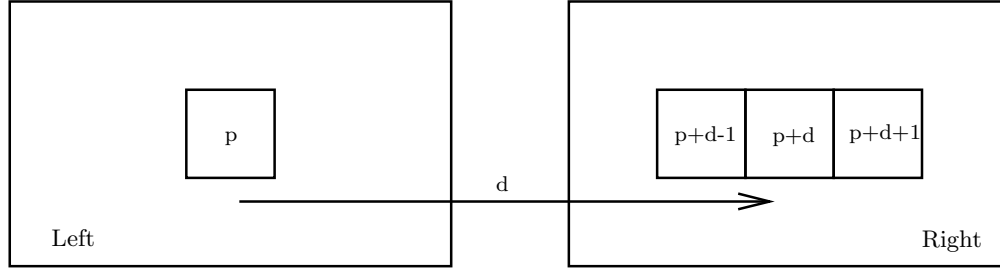


Figure 3.2: Insensitivity to image sampling for stereo graph cuts method where pixel p in the left image corresponds to pixel $p + \bar{d}$ in the right image.

$C_{fwd}(p, d)$ 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. Thus, a single data term is

$$D(p, d) = (\min\{C_{fwd}(p, d), C_{rev}(p, d), const\})^2 \quad . \quad (3.2)$$

where $const$ is a constant value that ensures robustness.

Even with the presence of the small negative effect of an increase in computation time, both design decisions are retained in this thesis.

3.1.2 The Smoothness Term

The original graph cuts for stereo algorithm chose a piecewise constant prior as their smoothness term E_{smooth} . Formally, given one-dimensional stereo disparity labels d_1 and d_2 , the smoothness term is defined as

$$V_{p,q}(d_1, d_2) = u_{p,q} \cdot (1 - \delta(d_1 - d_2)) \quad (3.3)$$

where

$$u_{p,q} = U(|I_p - I_q|) = \begin{cases} 2K & \text{if } |I_p - I_q| \leq \tau_I \\ K & \text{if } |I_p - I_q| > \tau_I \end{cases} \quad (3.4)$$

with K as our penalty constant and τ_I as our intensity difference threshold. τ_I is found experimentally with the optimal setting to 5. This function satisfies the three properties of a metric [8], formally defined as

$$\begin{aligned} V_{p,q}(\alpha, \beta) = 0 & \leftrightarrow \alpha = \beta && \text{[indiscernible]} \\ V_{p,q}(\alpha, \beta) & = V_{p,q}(\beta, \alpha) \geq 0 && \text{[symmetry]} \\ V_{p,q}(\alpha, \beta) & \leq V_{p,q}(\alpha, \gamma) + V_{p,q}(\gamma, \beta) && \text{[triangle inequality]} \end{aligned} \quad (3.5)$$

This allows the benefit of using the α -expansion algorithm to find a local minimum.

There are many benefits to using the α -expansion algorithm. First, there is a guarantee that this local minimum found is within a known factor of the global minimum. Second, this algorithm is $O(mn)$ where m is the number of nodes (pixels) and n is the number of labels (disparities). This accounts for a constant computation time per iteration (step 3 of Figure 2.12) given that the number of nodes in the graph are always constant.

However, this fact also leads to one downside of the approach. This algorithm is limited by the size of the images used and by the size of the disparity range. If we wish to have a larger disparity range computation time increases. Another problem with this algorithm is that it cannot handle more general cases of a smoothness function. For example, α -expansion is not appropriate in cases where the smoothness function becomes a semi-metric (satisfies only properties 1 and 2 from Equation 3.5).

3.2 Motion Estimation Using Graph Cuts

The big difference between stereo correspondence using graph cuts and motion estimation using graph cuts is that disparities are now two-dimensional. This addition of an extra

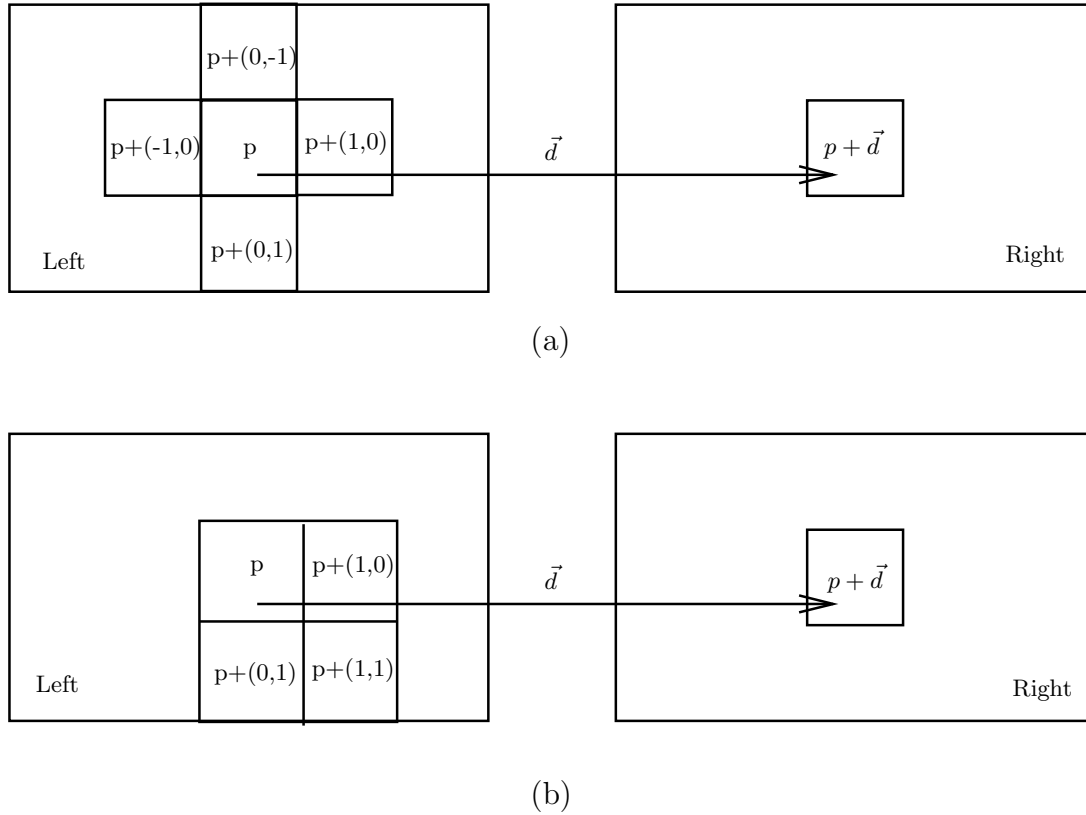


Figure 3.3: Image sampling insensitivity for (a) our motion graph cuts method and (b) the original graph cuts method where pixel p in the left image corresponds to pixel $p + \vec{d}$ in the right image.

dimension to the search space has large effects on the formulation of both the data and smoothness terms, as explained in the following subsections.

3.2.1 The Data Term

The data term is formulated in the same manner as the stereo algorithm in that it is insensitive to image sampling. However, this insensitivity is now over two-dimensions rather than a single dimension. Our implementation differs slightly from [8, 28]. We calculate insensitivity over the four neighbourhood system of a pixel (Figure 3.3 a) as opposed to the neighbouring lower right four pixel grouping of a pixel (Figure 3.3 b).

Given the two-dimensional disparity $\vec{d} = [d^h, d^v]^T$ for pixel p , we begin by measuring how well p fits into the two-dimensional range of disparities $(d_p^x - \frac{1}{2}, d_p^x + \frac{1}{2}) \times (d_p^y - \frac{1}{2}, d_p^y + \frac{1}{2})$.

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

where superscript h and v denote horizontal and vertical components, respectively. We get fractional values $I'_{p+(x,y)}$ by linear interpolation between discrete pixel values. For symmetry, we also measure

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

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

$$L = \min \{I_p, A_1, A_2, A_3, A_4, A_5\}$$

$$U = \max \{I_p, A_1, A_2, A_3, A_4, A_5\}$$

where

$$A_1 = \frac{I_p + I_{p+(-1,0)}}{2} \quad A_2 = \frac{I_p + I_{p+(1,0)}}{2} \quad A_3 = \frac{I_p + I_{p+(0,-1)}}{2} \quad A_4 = \frac{I_p + I_{p+(0,1)}}{2}$$

$$A_5 = \frac{I_p + I_{p+(-1,0)} + I_{p+(1,0)} + I_{p+(0,-1)} + I_{p+(0,1)}}{5} .$$

Therefore,

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

$C_{fwd}(p, d)$ can be computed similarly.

3.2.2 The Smoothness Term

The smoothness term chosen for motion estimation using graph cuts is that of a piecewise smooth prior. Formally, the interaction functional is

$$V_{mot_{p,q}}(d_p, d_q) = \lambda \min(const, (d_p^h - d_q^h)^2 + (d_p^v - d_q^v)^2) \quad (3.6)$$

where $const$ causes robustness, λ restricts the influence of the smoothness term and the superscript terms v and h denote the vertical and horizontal components of the motion vector, respectively.

This smoothness equation does not satisfy the triangle inequality property of 3.5 and as such, is a semi-metric. This forces us to utilize the more general α - β swap algorithm.

The swap algorithm is $O(mn^2)$ where m is the number of nodes (pixels) and n is the number of labels (disparities). At every iteration (step 3 of Figure 2.12), the number of graphs built is n^2 with each graph having a variable number of nodes because pixels labeled α or β make up the nodes rather than all the pixels in the image. As a result, computation time increases. Moreover, this algorithm no longer guarantees that the local minimum found is within a known factor of the global minimum, affecting accuracy.

3.3 Extending Graph Cuts to Use Combined Stereo and Motion Constraints

Combining both stereo and motion constraints provides tighter constraints on the system than either stereo or motion constraints used separately. In turn, this should have the desired effect of improving accuracy. Assuming the input to the system consists of four rectified images: $L_t, R_t, L_{t+1}, R_{t+1}$. These are, respectively, the left and right stereo pairs at time t and $t + 1$ of a stereo video sequence.

Refer to Figure 3.4 where we establish $Left_t$ as our reference frame. A point in our reference frame p_{L_t} is related to its corresponding points in the other images by the following four relationships

$$\begin{aligned}
 p_{L_t} + d_t &= p_{R_t} && \text{for stereo pair at time } t \\
 p_{L_t} + \vec{d}_L &= p_{L_{t+1}} && \text{for motion left pair} \\
 p_{L_{t+1}} + d_{t+1} &= p_{R_{t+1}} && \text{for stereo pair at time } t + 1 \\
 p_{R_t} + \vec{d}_R &= p_{R_{t+1}} && \text{for motion right pair}
 \end{aligned} \tag{3.7}$$

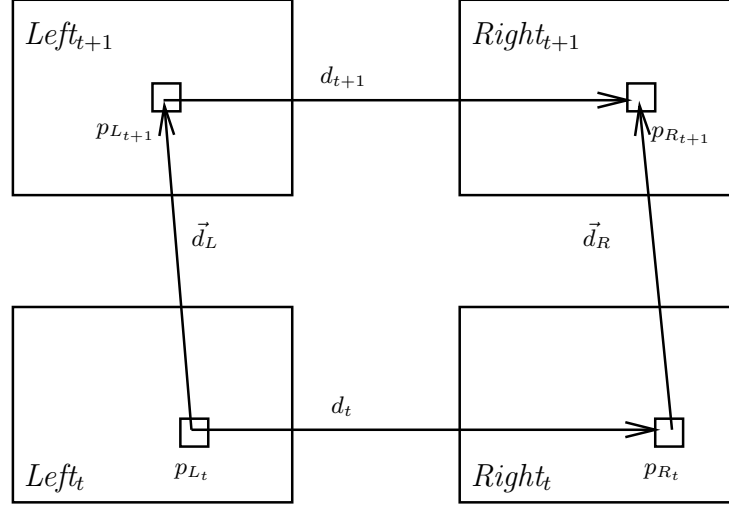


Figure 3.4: Example of two stereo image pairs $((L_t, R_t)$ and $(L_{t+1}, R_{t+1}))$ at times t and $t + 1$, respectively. A point p_{L_t} in the reference frame $Left_t$ is related to its corresponding points in the other images via established disparity values d_t , d_{t+1} , \vec{d}_L and \vec{d}_R for stereo pairs at time t , $t + 1$, left-motion image pairs, and right-motion image pairs, respectively.

where disparity values d_t , d_{t+1} , \vec{d}_L and \vec{d}_R are stereo pairs at time t , $t + 1$, left-motion image pairs, and right-motion image pairs, respectively. Thus, we can impose the following circular *combined stereo-motion* constraint.

$$0 = d_t + \vec{d}_R - d_{t+1} - \vec{d}_L \quad . \quad (3.8)$$

This combined set of disparities (2 stereo and 2 motion) creates a large label set. The number of labels is now equivalent to

$$|L| = st_x \times st_x \times (mot_x \times mot_y) \times (mot_x \times mot_y) \quad (3.9)$$

where st_x denotes the number of possible labels for stereo computation in the horizontal direction and mot_x and mot_y are the possible number of labels for motion in the horizontal and vertical directions, respectively. The $|\cdot|$ notation signifies the number of elements in the label set rather the absolute value function.

To index into this large six-dimensional set of labels, we have created a set of *super-labels*. This new set of labels constitutes a combination of the four possible disparities

as

$$f = \{f^{d_t}, f^{d_{t+1}}, \vec{f}^{d_L}, \vec{f}^{d_R}\} \quad (3.10)$$

where the superscript denotes the type of disparity. However, it is worth noting that given rectified images, it is possible to reduce this down to five dimensions. Once we obtain the vertical direction in one motion pair, we assume that the other motion pair has the same vertical disparity. This would reduce the number of labels to

$$|L| = st_x \times st_x \times (mot_x \times mot_y) \times (mot_x) \quad . \quad (3.11)$$

3.3.1 The Data Term

Adhering to the combined stereo-motion constraint, we require two stereo estimations and two motion estimations when choosing a data term for the global energy function. As a result, it was only natural to combine the previous data term equations for graph cuts calculation for stereo (from Section 3.1.1) or motion (from Section 3.2.1). The stereo terms are $C_{st}(p_{L_t}, f_p^{d_t})$ and $C_{st}(p_{L_{t+1}}, f_p^{d_{t+1}})$, and the motion terms are $C_{mot}(p_{L_t}, f_p^{\vec{d}_L})$ and $C_{mot}(p_{R_t}, f_p^{\vec{d}_R})$. Our resulting data term for pixel p given label f_p is formulated in a similar fashion to a L_2 norm.

$$D_p(f_p) = D_p(f_p^{d_t}, f_p^{d_{t+1}}, f_p^{\vec{d}_L}, f_p^{\vec{d}_R}) = \min(\tau_{D_{cutoff}}, A) \quad (3.12)$$

where

$$A = \sqrt{C_{st}(p_{L_t}, f_p^{d_t})^2 + C_{mot}(p_{R_t}, f_p^{\vec{d}_R})^2 + C_{st}(p_{L_{t+1}}, f_p^{d_{t+1}})^2 + C_{mot}(p_{L_t}, f_p^{\vec{d}_L})^2}$$

and $\tau_{D_{cutoff}}$ is a constant used to make the data term robust to outliers.

3.3.2 The Smoothness Term

We formulate the smoothness term $E_{smooth}(f)$ in our energy function 3.1 in a similar manner to what was done with the data term. Implementing our stereo-motion constraint

in our smoothness term, we arrive at

$$V_{p,q}(f_p, f_q) = \lambda \min(\tau_{S_{cutoff}}, B) \quad (3.13)$$

where

$$B = \sqrt{V_{st}(f_p^{d_t}, f_q^{d_t})^2 + V_{mot}(f_p^{\vec{d}_R}, f_q^{\vec{d}_R})^2 + V_{mot}(f_p^{\vec{d}_L}, f_q^{\vec{d}_L})^2 + V_{st}(f_p^{d_{t+1}}, f_q^{d_{t+1}})^2} \quad . \quad (3.14)$$

The constant $\tau_{S_{cutoff}}$ performs the same function as in the data term; providing robustness to outliers, while the λ term weights the influence of the smoothness term. The stereo interaction functions $V_{st}(f_p^{d_t}, f_q^{d_t})$ and $V_{st}(f_p^{d_{t+1}}, f_q^{d_{t+1}})$ are described in Equations 3.3 and 3.4 without the presence of their respective λ terms. The motion interaction functions are taken from Equation 3.6.

3.3.3 The Complete Energy Function

We combine both the data energy term and the smoothness energy term from the previous two subsections to obtain the global energy functional for combined stereo-motion. We represent this energy functional as

$$\begin{aligned} E(f_p) &= E_{data}(f) + E_{smooth}(f) \\ &= \sum_{p \in P} D_p(f_p) + \sum_{p,q \in N} V_{p,q}(f_p, f_q) \quad . \end{aligned} \quad (3.15)$$

3.4 System Overview

In this section, we present a brief overview of the proposed multi-resolution graph cuts using stereo-motion constraints system for visual correspondence. Figure 3.5 shows the high-level design of the system. Initially, input images enter the system at the *Pre-Processing* stage, which is designed to perform system initialisation. It receives the input images and ensures they are the same size, sets up logging functionality, loads input parameters from a scripting file and determines the number of possible labels in the label set.

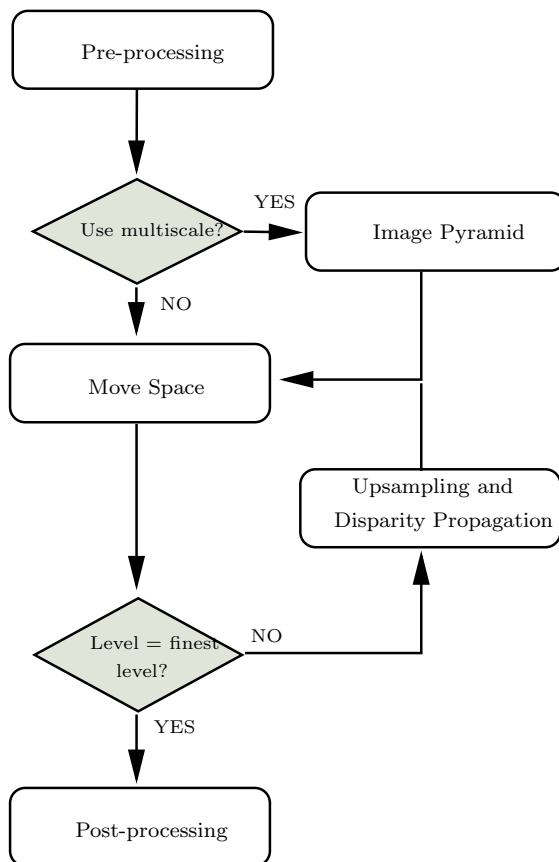


Figure 3.5: Flow diagram of multi-resolution graph cuts using stereo-motion constraints system.

Following the *Pre-Processing* stage are the key modules of the system. These stages deal with the implementation of the multi-resolution and graph cuts portions of the system. First is the *Image Pyramid* stage, which handles the creation of the input image pyramids. Second is the *Move Space* stage which builds the graphs that we find a minimum cut of. Finally, we perform an *Upsampling and Disparity Propagation* stage to upsample resulting disparity maps so that they help seed the next levels of the pyramid.

On the back end of the system, the *Post-Processing* stage handles all functions required for system shut down and results related activities. Resulting disparity maps and image pyramids are saved, while all parameters are reset for next set of tests.

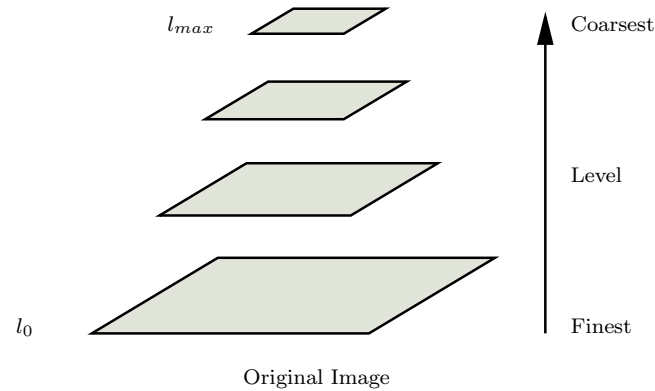


Figure 3.6: Example of an image pyramid. Level 0 is the original image at the finest scale, while l_{max} is the coarsest scale. Each level of the image is a blurred and downsampled version of the image at the previous level.

3.5 Image Pyramid Stage

The Image Pyramid stage computes Gaussian Pyramids of an image as depicted in Figure 3.6. The original image is at the bottom of the pyramid, at level l_0 . Each level of the pyramid represents a blurred and downsampled version of the image below it. We use a two-dimensional Gaussian filter, where advantage is taken of separability to enhance performance and to avoid aliasing.

Given a chosen sampling factor of 2, we require a Gaussian filter with a minimum radius of 2. In all our cases, we chose a σ of 2, resulting in a filter radius of 6. This ensures that no aliasing will occur in the image, giving the desired smoothing effect.

3.6 Move Space Stage

The *Move Space* stage builds many two-terminal graph structures that are minimized using a maximum flow algorithm attempting to find the configuration with the lowest amount of energy among all the possible labelings (Sections 2.5 and 2.6). Each two-terminal graph $G = \langle V, E \rangle$, with terminals α and β , contains a set of nodes $S =$

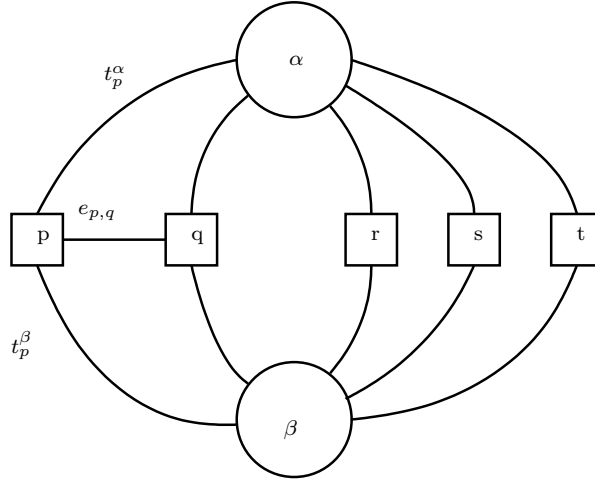


Figure 3.7: Example structure for a one-dimensional two-terminal graph G with a set of pixels p, q, r, s and t . Neighbouring pixels p and q are connected with a n-link $e_{p,q}$. Pixel p is connected to the terminals with t-links t_p^α and t_p^β .

$\{p | f_p \in \alpha, \beta\}$ that represent image pixels. Thus, the set of vertices is

$$V = \{\alpha, \beta\} \cup S \quad . \quad (3.16)$$

Edges in the graph are either t-links or n-links. t-links connect pixels p to terminals α and β and are represented as t_p^α and t_p^β , respectively. n-links $e_{p,q}$ connect neighbouring pixel pairs p and q (i.e. $\{p, q\} \in N$). Thus, the set of edges is

$$E = \left(\bigcup_{p \in S} \{t_p^\alpha, t_p^\beta\} \right) \cup \left(\bigcup_{\substack{\{p,q\} \in N \\ p,q \in S}} e_{p,q} \right) \quad . \quad (3.17)$$

An example of a one-dimensional graph, for the sake of legibility, is given in Figure 3.7.

3.6.1 Graph Building

Building the graph requires assigning weights to the edges. This process, however, is dependent on the structure of the smoothness term E_{smooth} . If the smoothness term is a metric, we can perform the α -expansion algorithm. On the other hand, if it is a semi-metric, we use the α - β swap algorithm to assign edge weights.

Expansion Algorithm

The expansion algorithm graphs contain two terminals, α and $\bar{\alpha}$. Therefore, the number of nodes in the graph corresponds to the number of pixels in the image. In the original papers on graph cuts [8, 28], the edge weights are assigned by inserting auxiliary nodes between neighbouring pixels containing different labels. These auxiliary nodes were later removed in [18] through a reformulation of the equations.

The idea behind the algorithm is that pixels are labeled as either α or $\bar{\alpha}$. After finding the minimum cut on the graph, pixels will either retain their previous labels or will change their label to α , whichever gives the least amount of energy in terms of the energy function. Since labels are only α or $\bar{\alpha}$, the algorithm need only pass through the label set once, determining the label configuration with the least amount of energy. Hence, the algorithm is $O(mn)$, where m is the number of pixels and n is the number of labels.

Swap Algorithm

Due to the semi-metric nature of the motion and combined stereo-motion smoothness terms discussed in Sections 3.2.2 and 3.3.2, we focus our attention on the α - β swap algorithm. Terminals in the graph are α or β . Only pixels labeled α or β make up the nodes in the graph. Therefore, a variable number of nodes are contained in each graph. Edge weights are assigned according to Table 3.1.

The idea behind the algorithm is that pixels are labeled as either α or β . After finding the minimum cut on the graph, pixels will either retain their previous labels or will swap their label to the other label, whichever gives the least amount of energy in terms of the energy function. In this manner, we need to compute every unique label combination in the label set. That way, the algorithm is $O(mn^2)$, where m is the number of pixels and n is the number of labels.

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

Table 3.1: Edge weight assignments for the α - β swap algorithm. (Reproduced from [28].)

3.6.2 Minimizing the Energy Function

In [5] they present a new algorithm which improves upon the performance of standard augmenting path techniques. Previous augmenting path algorithms use a breadth-first search when all paths of a given length are exhausted, requiring the construction of a new search tree. This process involves scanning the majority of the pixels in the image, becoming computationally expensive when repeated numerous times.

The new algorithm solves this problem by building two search trees, one from the source and another from the sink, and reusing them. They are never rebuilt from scratch. The result is an algorithm that is several times faster in all applications where graphs are 2D grids. However, this algorithm is not without its shortcomings. There is no guarantee that the augmenting paths found are necessarily the shortest ones.

The idea behind the algorithm is to maintain two search trees of nodes interconnected by edges in order to find an augmented path AP from the source tree's root node to the sink tree's root node, as illustrated in Figure 3.8. The two search trees S and T begin with their root nodes at the source s and sink t , respectively. In tree S all edges from the parent node to its children are non-saturated, while in tree T edges from children to their parents are non-saturated.

The algorithm allows for three types of nodes: (1) free, (2) active A , and (3) passive P nodes. Nodes that are not in tree S or T are free nodes. Active nodes are the border

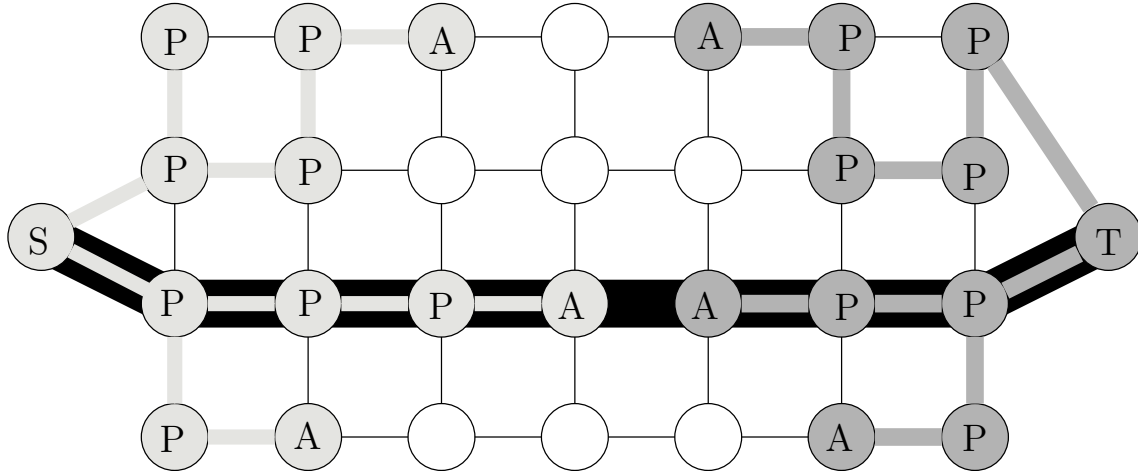


Figure 3.8: Example of the search trees S (light grey) and T (dark grey) containing active A and passive P nodes. All remaining nodes, free nodes, are empty (white) nodes. The resulting augmenting path P found is in black. (Reproduced from [5].)

nodes of each tree that allow tree growth by acquiring new children from a set of free nodes. Passive nodes are the internal nodes that were previously active nodes.

The improved augmenting path algorithm is summarized in Figure 3.9. This algorithm iteratively processes three stages: (1) growth, (2) augmentation and (3) adoption stages. During the growth stage, both trees try to grow by acquiring nodes from a set of free nodes. A tree's set of active nodes explores adjacent non-saturated edges acquiring new children from the set of free nodes, which become active nodes. After having explored all neighbouring nodes, an active node becomes passive, indicating an inability to grow further. The growth stage terminates when an active node encounters an active node from the other tree, signifying the creation of an augmenting path AP .

The augmentation stage augments the path found with the largest flow possible, sometimes creating saturated edges and subsequently orphan nodes. Meanwhile, the adoption stage finds parents for each orphan, ensuring that the parent be connected through a non-saturated edge and belong to the same tree as the orphan. If we are unable to find a parent node, the orphan node becomes a free node. This stage ends when there are no more orphan nodes.

```

initialize       $S = s, T = t, A = \{s, t\}, O = 0$ 
WHILE true
    grow  $S$  or  $T$  to find an augmenting path  $AP$  from  $s$  to  $t$ 
    IF no more tree growth possible, terminate
    augment path  $AP$ 
    adopt orphans in  $O$ 
END while

```

Figure 3.9: Outline of the improved augmenting path algorithm. (Reproduced from [5].)

The algorithm runs until the search trees S and T can no longer grow, indicated by no more active nodes, and the trees are separated by saturated edges. This is the point where maximum flow is reached. The resulting minimum cut is the border between S and T where all edges are saturated.

3.7 Upsampling and Disparity Propagation Stage

For the multi-resolution algorithm to work properly, disparity maps must propagate between levels of the pyramid. These disparity maps seed the next level of the pyramid, thereby giving a good initialisation point for the algorithm. Going from a coarser level to a finer level of the pyramid, upsampling takes every pixel p and creates $N \times N$ pixels, as depicted in Figure 3.7. The disparity values for the pixels with known disparities at the coarser scale are multiplied by the sampling factor used. However, there is a large set of $N \times N - 1$ new pixels that have no known label. Typically, algorithms perform some type of linear or bilinear interpolation is done on the labels between pixels with known labels to better approximate the values. However, this process adds an unnecessary computational expense to the system given that the disparity map is generally piecewise constant and

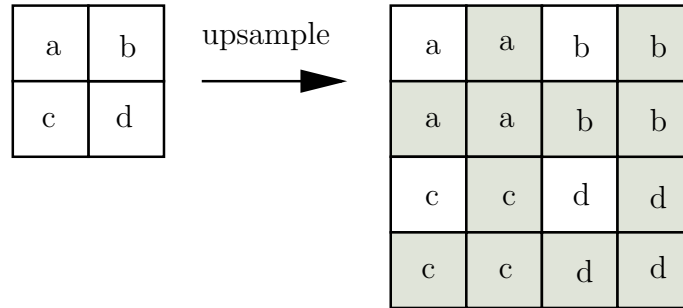


Figure 3.10: Upsampling from a finer level l_i to a coarser level l_{i-1} of the pyramid. Pixels with known labels at l_i are multiplied by the sampling factor $k = 2$ to determine the labels at l_{i-1} . Pixels with unknown labels are in grey. These pixels are assigned the same label as their neighbour pixel that originated from the same group.

that graph cuts algorithm already computes all label combinations. Consequently, pixels with unknown labels are assigned the label of the pixel that they were upsampled from.

3.8 Various Multi-Resolution Graph Cuts (MRGC) Algorithms For Stereo-Motion

As is common in most stereo algorithms, there is a trade-off between computation time and accuracy. Imposing tighter constraints on the system and increasing the number of possible disparities are ways to improve accuracy. However, in the graph cuts framework this change causes the undesirable effect of increasing computation time. Inversely, computing on smaller label sets or reducing the number of pixels in the image are ways to reduce computation time but also reduce system accuracy. The balance between these two factors is an important design consideration when developing an algorithm.

Each algorithm in this section is a multi-resolution graph cuts (MRGC) algorithm. The benefits of a multi-resolution approach to graph cuts are:

1. **Reduced number of pixels:** The image dimensions are reduced by the sampling factor k at each level of the pyramid. This results in the number of pixels being reduced by a factor of k^2 .
2. **Reduced number of labels:** Each dimension of the disparity range is reduced by the sampling factor k at each level of the pyramid. Consequently, for image pairs, the number of stereo labels decreases by k labels at each level of the pyramid, while the number of motion labels decreases by k^2 . Thus, using our four image set of two stereo-motion image pairs, the number of labels decreases by k^6 .
3. **Improved computation time:** Since each level has a smaller number of pixels and labels, convergence at each level is faster. Similarly, the disparity maps found at each level of the pyramid allow initialisation of the next level's label configuration closer to that level's optimal minimum, resulting in the same effect. By the time we reach the finest level of the pyramid, the configuration is closer to the overall global minimum than the normal graph cuts initialisation for stereo-motion method. Therefore, we reach convergence at a faster rate, reducing computation time in the process.

The general form of the multi-resolution approach is described in Listing 1, called the *Level Seeding MRGC (LS)* algorithm. $Dmap_i$ represents the disparity map obtained for level i with $Dmap_{final}$ referring to the final solution obtained at the finest level. To determine the disparity range for each level, we take the maximum disparities allowed in both the vertical and horizontal directions and divide by the sampling factor. For example, if the maximum disparity range is [16,16] for motion disparities at the finest level (level 0), it becomes [8,8], [4,4] and [2,2] for levels 1, 2 and 3 respectively. The energy minimisation algorithm (EMA in step 3c) is the step that varies among all the algorithms described in this section. However, in this general level seeding approach steps 3a and

```

1. Create Gaussian image pyramid
2. Determine disparity range for each level of the pyramid
3. FOR i = numLevels-1 to 0
    IF i = (numLevels-1)
3a. Dmap_i = Normal stereo-motion graph cuts algorithm
    ELSE
3b. Upsample disparity map
3c. Dmap_i = EMA(Dmap_{i+1})
4. Dmap_final = Dmap_0

```

Listing 1: General outline of all multiscale methods. $Dmap_i$ represents the disparity map obtained for level i , while EMA refers to the energy minimization algorithm used in computation.

3b do not vary. This approach has the property that each level of the pyramid seeds, or initialises, the next finer level of the pyramid.

The rest of this section describes three algorithms developed using the MRGC framework, each of which attempts to reduce the number of labels used during the EMA in step 3c. We pay more attention to the α - β swap algorithm given that this is the necessary algorithm (since our objective function is a semi-metric) for our combined stereo-motion framework.

3.8.1 Label Disparity Neighbourhood Restricted MRGC (LDNR)

The key problem with the LS algorithm described above is that despite its beneficial level initialisation ability, the label set is too large and as such, computation time becomes very large. To remedy this problem, we have developed an algorithm, called *Label Disparity Neighbourhood Restricted MRGC (LDNR)*, that uses the notion of *label disparity neighbourhoods* to restrict the size of the label sets.

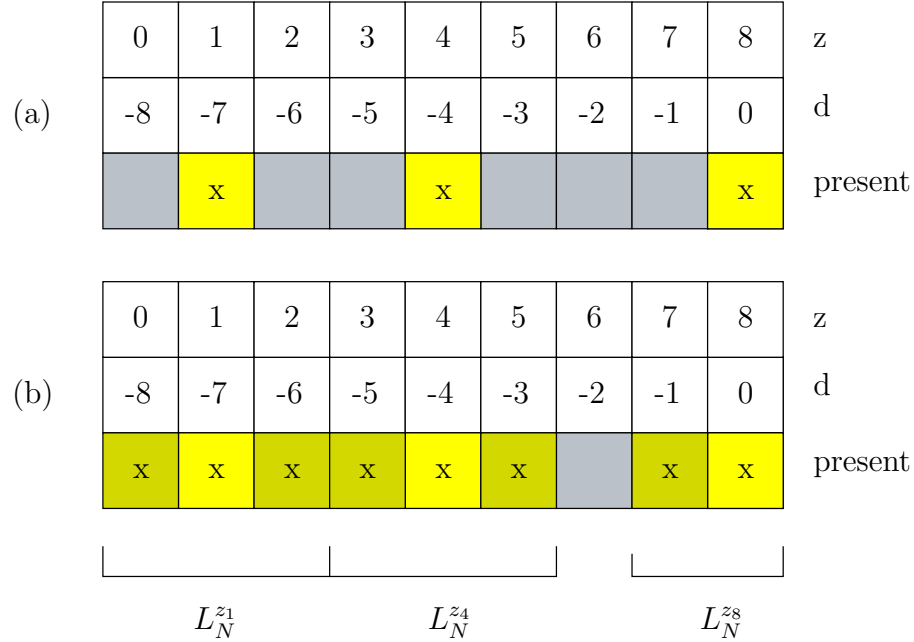


Figure 3.11: The (b) label disparity neighbourhood of (a) the labels present (in yellow).

A label disparity neighbourhood is defined as the set of label values that are within a specified distance *range* of the disparity value of the current label z_1 . The labels cannot have disparity values that are outside the allowable disparity range. The value of *range* can be any scalar value desired, but in general we set *range* to 1. To help illustrate this, we use a one-dimensional view of a sample label set L in Figure 3.11, where the neighbourhood of label z_1 with disparity value d is represented as $L_N^{z_1}$. For example, the label z_1 corresponds to a disparity value $d = -7$ with a label disparity neighbourhood $L_N^{z_1} = [z_0, z_1, z_2]$. For motion labels, where disparities are two-dimensional, we take the same approach but apply it in both dimensions. Therefore, the *range* = 1 size of the label disparity neighbourhood is 9 as opposed to a maximum size of 3 for stereo labels. For stereo-motion labels, this translates into a maximum size of $3 \times 3 \times 9 \times 9 = 729$ labels.

The idea behind this algorithm is to allow pixels with label z_1 to swap labels with pixels having a label in the label disparity neighbourhood $L_N^{z_1}$ of z_1 . This decreases the

number of α - β swaps from all label combinations to only combinations of a label and those within its label disparity neighbourhood. For example, in the above case the disparity range is $[-8, 0]$, which gives 9 different labels and 81 label combinations (or α - β swaps). However, we only require unique label combinations (due to the symmetry property of metric Equation 3.5). Therefore, we only require the upper triangle of the 9×9 label combinations matrix minus the diagonal, resulting in 36 unique label combinations. If the labels present were $L_{present} = [z_1, z_4, z_8]$, indicated in yellow, then the allowable label combinations are:

1. **For** $L_N^{z_1}$: $(z_1, z_0), (z_1, z_1), (z_1, z_2)$
2. **For** $L_N^{z_4}$: $(z_4, z_3), (z_4, z_4), (z_4, z_5)$
3. **For** $L_N^{z_8}$: $(z_8, z_7), (z_8, z_8)$.

We are left with 8 label combinations. If we take into account that swapping a label with itself does nothing, we are down to 5 label combinations. Going from 36 unique label combinations to 5 is a large reduction of α - β swaps to perform, thereby decreasing the computation time.

The justification for reducing the label combinations to only the label disparity neighbourhoods is that pixels label are assumed to already be close to their ideal label. Swapping labels within their label disparity neighbourhood allows the refinement of the disparity estimates, where labels travel a short distance. Over multiple iterations, however, a pixel's label can change by a value greater than *range*.

The implementation of LDNR uses the algorithm described in Listing 1 where the energy minimisation algorithm EMA in step 3c is shown in Listing 2. Input to the algorithm is the upsampled disparity map from the previous level of the pyramid. The convergence condition tests if the current iteration's energy is less than the previous iteration's energy. This energy value is determined by computing the current configuration

```

EMA for LDNR: INPUT = Dmap_up
-----
1. Determine label set  $L_{\text{present}}$  in Dmap_up
2. WHILE  $E_{\text{cur}} < E_{\text{prev}}$ 
2a.  $E_{\text{prev}} = E_{\text{cur}}$ 
2b. Perform one iteration of SWAP
    FOR each label  $l_a$  in  $L_{\text{present}}$ 
        Determine label neighbourhood  $L_N$  of  $l_a$ 
        FOR each label  $l_b$  in  $L_N$  of  $l_a$ 
            Dmap_cur = SWAP( $l_a$ ,  $l_b$ )
2c. Compute  $E_{\text{cur}}$  from Dmap_cur
3. SET Dmap_final = Dmap_cur

```

Listing 2: LDNR energy minimization algorithm.

of the disparity map with the global energy function. If the function has not converged, the algorithm begins by determining all the labels present L_{present} in the upsampled disparity map from the previous level (step 1). This label set does not change throughout the entire operation of EMA. Then it performs one iteration of SWAP in step 2b. Recall from Figure 2.12, that an iteration computes the configuration with the least amount of energy among all label combinations within one move space of the current configuration. However, in the LDNR algorithm, this number of combinations is restricted. An iteration in this algorithm iterates through the labels in L_{present} . For each one of these labels l_a , it finds its label disparity neighbourhood $L_N^{l_a}$ and performs α - β swap between l_a and every label l_b in $L_N^{l_a}$.

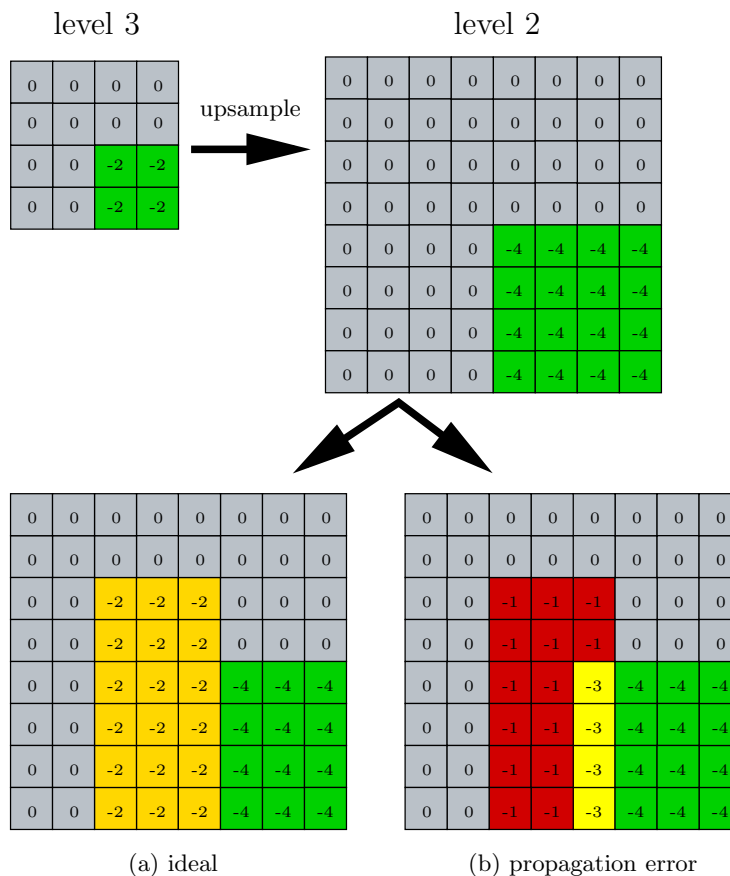


Figure 3.12: After upsampling a disparity map from level 3 to level 2, we find the presence of label error propagation between levels of the pyramid. The ideal solution for level 2 should be (a). However, a disparity value of -2 is not in the label disparity neighbourhood of disparity values of 0 or 4. Therefore, using LDNR, the disparity map achieved at level 2 would be as in (b) where the disparity values of -1 (red) and -3 (yellow) are errors that propagate to subsequent levels.

3.8.2 Expanding Label Disparity Neighbourhood at Every Iteration MRGC (EL)

The decrease in the number of labels is the main benefit of the LDNR algorithm. However, with such a large decrease, the accuracy will suffer. Therefore, we consider a second algorithm that accounts for the problem of reducing the label set size by too large a margin. This algorithm is called *Expanding Label Disparity Neighbourhood at Every Iteration MRGC* (EL).

The cause of the loss in accuracy in the previous algorithm is due to the error propagation of invalid disparity values between levels of pyramids. This is due to the combination of the upsampling function and the restriction that labels are only allowed to swap within their own label disparity neighbourhood. See Figure 3.12 as a helpful illustration of this problem. Figure 3.12 (a) represents the ideal solution with Figure 3.12 (b) showing the disparity error propagation. At level 3, the disparity maps are found. They are upsampled with the disparity values doubling and propagating to neighbouring pixels to get the disparity maps at level 2. The problem begins with the newly created pixels. Some of these pixels should be assigned a disparity value other than what they currently have; for example, the red and yellow pixels in (b). However, these pixels are restricted to only swap with labels in their label disparity neighbourhood, thereby never getting properly assigned (pixels in orange in (a)). This error will then propagate to the next level and the next, until we reach the finest level of the pyramid where the assigned disparity values are no longer near their ideal values.

To remedy this problem, we allow the label set $L_{present}$ to grow at every iteration, shown inside step 2b of the EL algorithm in Listing 3. This label set remained constant in the LDNR algorithm, restricting labels to swap only with labels in its label disparity neighbourhood. By moving this step inside step 2b, we allow the label set to grow slowly. Label swaps are still done between a label present in $L_{present}$ and its label disparity neighbourhood, but over time the disparity values slowly propagate towards their ideal values. With this increased set of label swaps, the accuracy will increase, but computation time increases.

3.8.3 Swap All Combinations LDNR-MRGC (SAC)

Another method to improve accuracy is to allow the swapping of more label combinations. In the LDNR algorithm, α - β swaps were restricted to occur between a label and the labels

```

EMA for EL: INPUT = Dmap_up
-----
1.  Dmap_cur = Dmap_up
2.  WHILE E_cur < E_prev
2a.  E_prev = E_cur
2b.  Perform one iteration of SWAP
      Determine label set L_{present} in Dmap_cur
      FOR each label l_a in L_{present}
        Determine label neighbourhood L_N of l_a
        FOR each label l_b in L_N of l_a
          Dmap_cur = SWAP(l_a , l_b)
2c.  Compute E_cur from Dmap_cur
3.  SET Dmap_final = Dmap_cur

```

Listing 3: EL energy minimization algorithm.

in its label disparity neighbourhood. The new algorithm, called *Swap All Combinations LDNR-MRGC* (SAC), removes this restriction.

The algorithm for SAC is found in Listing 4. Similar to LDNR, it determines the labels present $L_{present}$ in the upsampled disparity map (step 1a), without growing throughout the algorithm. The next step involves determining the label disparity neighbourhoods for each of the labels in $L_{present}$, adding these labels to the label set L_{todo} (step 1b). This new label set would correspond to the third row, “present”, of Figure 3.11. Then it computes the α - β swaps for each of the label combinations of labels in L_{todo} in step 1c.

In the case of stereo algorithms or motion algorithms, the label set could conceivably become the full label set. In stereo-motion algorithms, however, the label set never reaches the full label set. This is due to the impossibility of some disparity combinations in the super labels. For instance, a super label $f = \{f^{d_t}, f^{d_{t+1}}, \vec{f}^{d_L}, \vec{f}^{d_R}\}$ with values $f = [-16, 0, (-16, 0), (0, 0)]$ is unlikely to be supported by the data. This pixel has two

```

EMA for SAC: INPUT = Dmap_up


---


1. WHILE E_cur < E_prev
1a. Determine label set L_{present} in Dmap_up
1b. FOR each l_i in L_{present}
    Determine label neighbourhood L_N of l_i
    Add L_N of l_i to label set L_todo
1c. Perform one iteration of SWAP
    FOR each label l_a in L_todo
        FOR each label l_b in L_todo
            SWAP(l_a , l_b)
2. SET disparity map = current configuration

```

Listing 4: SAC energy minimization algorithm.

very different stereo disparities and two very different motion disparities. Nonetheless, there is a large increase in label combinations, causing a large increase in computation time. On the other hand, we also receive a valuable increase in accuracy.

3.9 Summary

The design of a multi-resolution graph cuts algorithm for stereo-motion requires deciding what data and smoothness functions to use. The choice of a smoothness function has a large influence on the computational expense of the system. A metric smoothness function allows the use of the faster α -expansion algorithm in the graph building stage, with a time complexity of $O(mn)$ where m is the number of pixels and n is the number of labels. However, if only a semi-metric smoothness function is possible, the use of the α - β swap algorithm $O(mn^2)$ is required.

Given the known time complexity functions, reducing the number of pixels and the number of labels is key to a faster algorithm. We used a multi-resolution approach to accomplish this task, whereby a graph cuts algorithm for stereo-motion was performed at each level of the pyramid, then seeding the next level of the pyramid. Seeding initialised each level closer to its optimal solution, thereby improving computation time.

We also developed three different algorithms (LDNR, EL, SAC) that reduced the number of labels. In the next chapter, we look at the results of these algorithms and the overall system performance. In particular, we examine the trade-off between system accuracy and computation time.

Chapter 4

Results

This chapter evaluates the performance of the *Multi-Resolution Graph Cuts for Stereo-Motion* system in comparison to the original *graph cuts for stereo* technique of [6, 8, 5, 18, 28]. The system implementation was done in C++ on a Pentium IV 3GHz machine with 4GB of memory, running the Linux operating system. Table 4.1 shows the system input parameters used to generate the results in this chapter. Input parameter representation can be found with their associated equation, if applicable, shown in the third column. The disparity range and the border size is divided by the downsampling/upsampling factor for each level of the pyramid. For example, at level 0 of the pyramid, the border is 18 pixels and the horizontal disparity range is [-16,0], while at level 1, the border is $18 \div 2 = 9$ pixels and the horizontal disparity range is [-8,0].

Without the presence of any known ground truth data for stereo image sequences with motion, we are forced to test all algorithms against ground truth data obtained from the Middlebury Stereo webpage [22]. All image sequences are rectified stereo image sequences that contain static scenes and robust objects. We test our algorithms using the Tsukuba sequence, which is composed of five colour images of size 384×288 pixels, shown in Figure 4.1. The camera motion is a simple translation from right to left. Ground truth

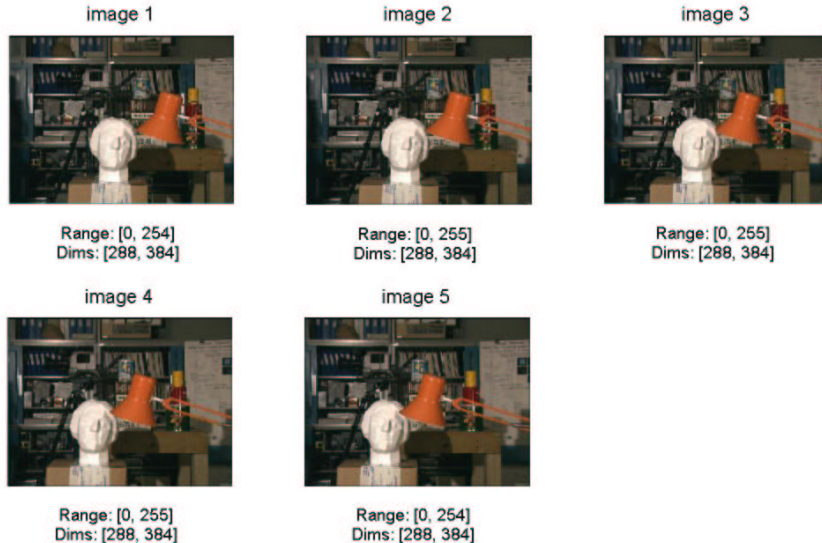


Figure 4.1: Tsukuba image sequence composed of 5 colour images of size 384×288 pixels.

data is also provided, accompanied by an occlusion map, a depth discontinuity map and a textureless region map, which are shown in Figure 4.2.

Our stereo-motion framework is tested using three consecutive images of the sequence. For example, the first stereo pair of images L_t and R_t correspond to images 3 and 4 from Figure 4.1. The second stereo pair of images at time $t + 1$, L_{t+1} and R_{t+1} , correspond to images 4 and 5. Thus, the disparity value d_1 corresponds to stereo pair at time t (images 3 and 4), while d_2 corresponds to stereo pair at time $t + 1$ (images 4 and 5). The motion disparity values d_L and d_R correspond to the left motion pair (images 3 and 4) and the right motion pair (images 4 and 5), respectively. Therefore, our images exhibit no vertical motion due to the images being rectified.

The rest of this chapter is organized as follows. Section 4.1 illustrates the Gaussian pyramids created for the image sequence. Section 4.2 analyses the influence that label set size has on the computation time of the system. Finally, Section 4.3 looks at the disparity maps generated by each algorithm presented in Chapter 3 and compares accuracy against computation time.

Parameter	Value	Equation
Disparity range:		
$[x_{min}, x_{max}]$	$[-16, 0]$	
$[y_{min}, y_{max}]$	$[0, 0]$	
Border size (pixels)	18	
Number of pyramid levels	4	
Down/upsampling factor	2	
Label disparity neighbourhood range	$+/- 1$	
$\tau_{D_{cutoff}}$	1000	3.12
τ_I	5	3.4
$\tau_{S_{cutoff}}$	1000	3.13
λ	20	3.13
K	1	3.4

Table 4.1: Values of the parameters used to compute the results in Chapter 4.

4.1 Gaussian Pyramids

The first major stage in the flow diagram of Figure 3.5 is the *Image Pyramid* stage described in Section 3.5. This stage involves recursively blurring an image and then downsampling it to get the next coarser level of the pyramid. For all algorithms implemented in this paper, we used $\sigma = 2$ for the Gaussian filters and we downsampled the images by a factor of two.

Both the MRGC stereo algorithms and the MRGC motion algorithms generate Gaussian pyramids for two input images. Figure 4.3 shows a sample of these Gaussian pyramids generated by the *Image Pyramid* stage. In the pyramid, the original input images are shown at level 0. For stereo, *image 1* and *image 2* correspond to images $Left_t$ and $Right_t$, respectively. When computing motion, *image 1* and *image 2* correspond to the

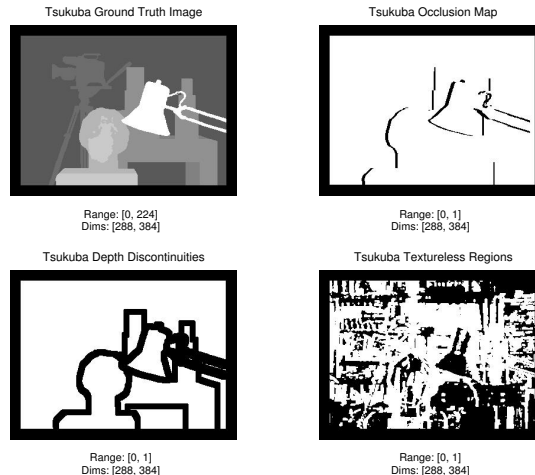


Figure 4.2: Ground truth disparity map (top left), occlusion map (top right), depth discontinuity map (bottom left) and textureless region map (bottom right) for the Tsukuba image sequence.

im_t and im_{t+1} image pairs, respectively. The border size for each of the levels, starting at level 0, is $[18,9,4,2]$, with the axes corresponding to image size.

Figure 4.4 shows an example Gaussian pyramid generated by the combined stereo-motion algorithms described in Section 3.8. The input images for these algorithms are shown at level 0 with the images $Left_t$, $Right_t$, $Left_{t+1}$ and $Right_{t+1}$ corresponding to the two stereo image pairs from Figure 3.4.

4.2 Number of Labels

One of the most important measures of any system is its computation time. In the graph cuts framework, factors influencing computation time are the number of pixels m and the number of labels n . This is evident in the time complexity equations $O(mn)$ for the α -expansion algorithm and $O(mn^2)$ for the α - β swap algorithms.

Figures 4.5 and 4.6 show the results of varying the number of labels and their effect on processing time. The number of labels represent the size of the allowable disparity range or the number of possible disparity values that a pixel can be assigned. Figure 4.5

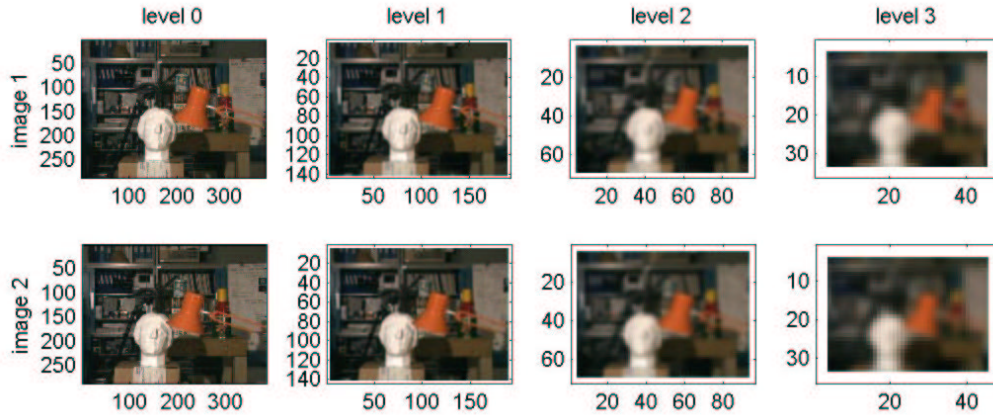


Figure 4.3: Two images are used to create the image pyramids above and are used as input to the stereo (image 1 = $Left_t$ and image 2 = $Right_t$) or motion (image 1 = im_t and image 2 = im_{t+1}) graph cuts algorithms. From left to right, the images proceed up the pyramid, passing from the finest to the coarsest level of the pyramid with each image axis indicating the image dimensions in pixels. The border size for each of the levels, starting at level 0, is $[18,9,4,2]$, with the axes corresponding to image size.

compares the original stereo graph cuts algorithm to the LS stereo algorithm. Figure 4.6 does likewise, but compares results between the standard motion graph cut algorithm and the LS motion version. As seen in the figures, there is a direct correlation between the number of pixels and the computation times of each algorithm. As the number of labels increases, the computation time increases. Everything to the left of the vertical dotted lines appears to follow the time complexity equations for their respective graph building algorithm. During computation, we only compute unique label combinations, which explains the near linear curve of α -expansion $O(mn)$ for stereo. For motion, it appears to follow $O(mn^2)$ for the α - β swap algorithm. However, no regression analysis has been done to confirm these interpretations. Also, we should note that everything to the right of the vertical lines appears linear. This is explained by the fact that there are only 16 possible labels attainable in the Tsukuba sequence. As such, extra computation is added through a loop searching for pixels assigned the extra labels. Since there is an absence of pixels assigned these extra labels, no time is spent building a graph and finding the minimum cut. Also, another important result is that the LS algorithm for motion

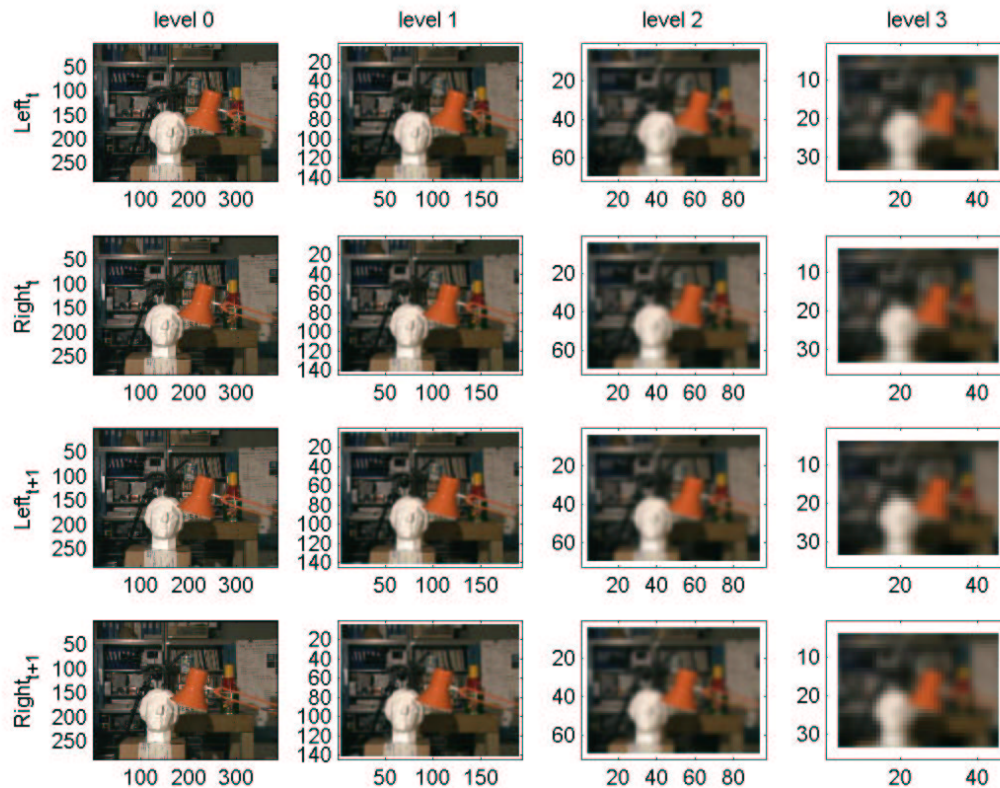


Figure 4.4: Four images are used to create the image pyramids above and are used as input to the combined stereo-motion graph cuts algorithms. From top to bottom, the images correspond to the $Left_t$, $Right_t$, $Left_{t+1}$ and $Right_{t+1}$ of the two stereo pairs in Figure 3.4. From left to right, the images proceed up the pyramid, passing from the finest to the coarsest level of the pyramid with each image axis indicating the image dimensions in pixels.

starts to out-perform the standard motion graph cuts algorithm in terms of computation time once the label set sizes start to become large.

Table 4.2 and Figure 4.7 also demonstrate this characteristic for all algorithms implemented, with Figure 4.7 being the graphical representation of the data in Table 4.2. The term “Normal” in Table 4.2 refers to straightforward computation of the specific method. The number of labels designate the final number of labels used for computation during the final iteration before convergence. Despite the difference in the number of labels, all rows with the same method solve the mentioned problem. In Figure 4.7, we can see

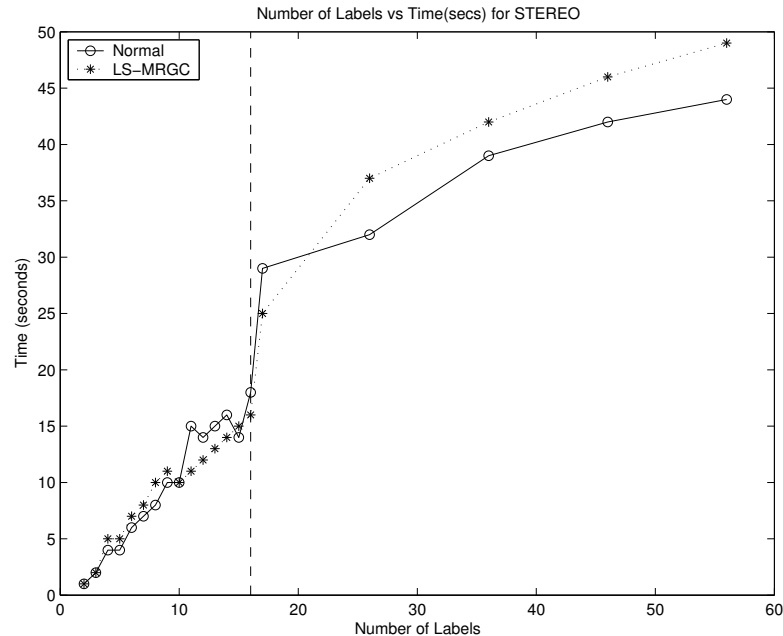


Figure 4.5: The relationship between the number of labels and the time to convergence for the original stereo graph cuts algorithm and the LS stereo graph cuts algorithm. The dotted vertical line indicates the range of disparities known to be actually present in the data.

that we exhibit near linear computation time for stereo graph cuts algorithms, while we do better than quadratic computation time for the motion and combined stereo-motion algorithms.

At the time of writing, using both the stereo-motion graph cuts algorithms “Normal” and LS, the number of labels is too large. As a result, the experiments are still ongoing. We estimate these algorithms to reach convergence in approximately 6 months, based on interpolating the time it took for one iteration to complete. Thus, the value “A” in the table is a very large, while the value “B” for LS for stereo-motion graph cuts is also large, but most likely a number less than “A”. Using a multi-resolution framework initializes the energy function closer to the minimum, thereby achieving convergence faster.

Figure 4.8 illustrates the influence of the number of pixels and the number of labels on computation time. An iteration in the graph indicates when the algorithm has created all possible label combinations, calculated the minimum energy of each graph for each

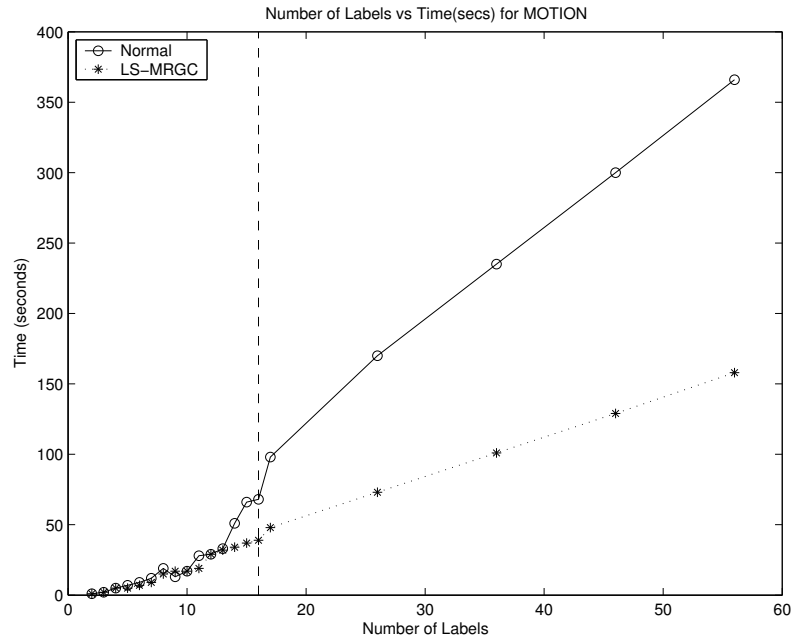


Figure 4.6: The relationship between the number of labels and the time to convergence for the motion graph cuts algorithms. The dotted vertical line indicates the range of disparities known to be actually present in the data.

of these label combinations, and produced the configuration with the lowest amount of energy. The graph shows a plot of iterations on the x-axis versus the time to completion (seconds) on the y-axis. The vertical dotted lines in the graph indicate when a level change has occurred in the algorithm. Going from left to right, these changes occur at iterations 7 (from level 3 to 2), 15 (from level 2 to 1) and 25 (from level 1 to 0).

Level changes have two effects on the system: (1) the image dimensions increase by the sampling factor (factor = 2 in all our cases); and (2) the disparity range increases by this same factor. Therefore, the number of pixels increases by a factor of $2^2 = 4$ at each level change. Similarly, the number of labels increases by a factor of 2 for α -expansion and by a factor $2^2 = 4$ for α - β swap. The resulting effect is an increase in computation time. This effect is visible in the increased computation time as we pass from a higher (coarser) level of the pyramid to a lower (finer) level of the pyramid where the slope increases for each level.

ID No.	Method	Algorithm	No. Labels	Time (secs)
1	Stereo	Normal	17	23
2	Stereo	LS	17	23
3	Stereo	LNR	17	23
4	Stereo	ELNEI	17	23
5	Motion	Normal	17	99
6	Motion	LS	17	48
7	Motion	LNR	7	7
8	Motion	ELNEI	14	11
9	Stereo-motion	Normal	83521	A
10	Stereo-motion	LS	83521	B
11	Stereo-motion	LNR	251	294
12	Stereo-motion	ELNEI	2890	2073
13	Stereo-motion	SAC-LNR	10798	533593

Table 4.2: The relationship between the number of labels and the time to convergence for all graph cuts algorithms. A is a very large time (estimated as approximately 6 months) and B is also large and is less than A.

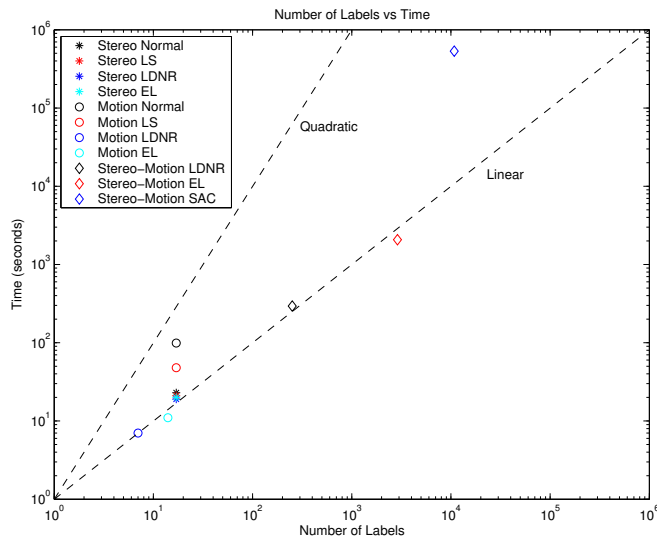


Figure 4.7: The relationship between the number of labels used in an algorithm and the time to convergence for all graph cut algorithms (stereo, motion and combined stereo-motion).

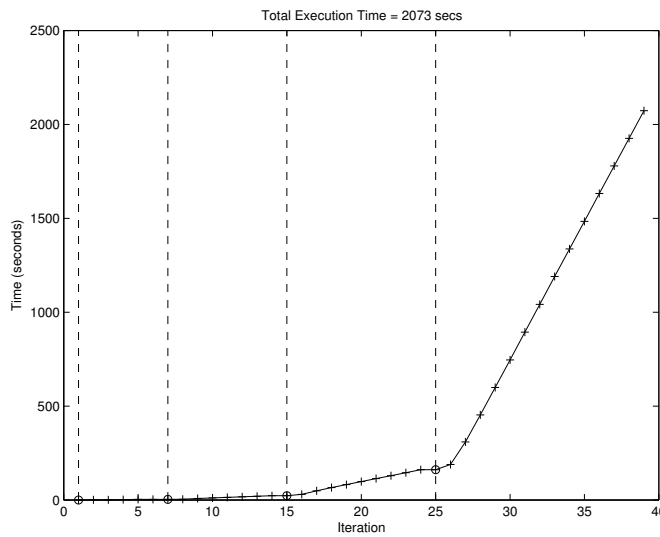


Figure 4.8: Sample computation time graph for a multi-resolution graph cuts (MRGC) algorithm. Vertical dotted lines indicate a change in level operation in the MRGC algorithm. The first dotted line on the left begins at the highest (coarsest) level of the pyramid operation, finishing with the lowest (finest) level of the pyramid. This example begins at level 3 and finishes at level 0.

4.3 Accuracy

To provide a quantitative measure of system accuracy, we compare the disparity maps obtained from each algorithm to the ground truth data provided by the Middlebury Stereo webpage [22], shown in the top left image in Figure 4.2. Accuracy is computed according to two measures:

1. **Root mean square error (RMSE)**: Measured in pixel disparity units between the estimated disparity map $d_{est}(x, y)$ and the ground truth disparity map $d_{gt}(x, y)$.

This is represented as

$$RMSE = \sqrt{\frac{1}{N} \sum_{(x,y)} |d_{est}(x, y) - d_{gt}(x, y)|^2} \quad (4.1)$$

where N is the total number of pixels.

2. **Accuracy**: The percentage of pixels that have the same disparity value as the ground truth data. We represent this as

$$Accuracy = \left(\frac{1}{N} \sum_{(x,y)} (|d_{est}(x, y) - d_{gt}(x, y)| < \delta_d) \right) \times 100 \quad (4.2)$$

where δ_d is a disparity error tolerance. We set δ_d to 1.0 pixels.

Disparity values are only considered in regions of non-occlusion. Thus, we mask the output disparity maps with the occlusion map in the top right image of Figure 4.2, where white pixels indicate regions of non-occlusion. Error histograms are used to illustrate the distribution of absolute value of the disparity error in pixels. Energy over time plots are used to show the measure of accuracy of a solution. Lastly, we also provide maps indicating the pixels the labels that have been labeled inaccurately.

The rest of this section is organized as follows. First, we show all disparity maps obtained for each of the algorithms: stereo graph cuts algorithms, motion graph cuts algorithms and combined stereo-motion graph cuts algorithms. Second, we analyze the

accuracy of each of the algorithms using the previously mentioned metrics. More importantly, we compare the trade-off between the system accuracy against computation time.

4.3.1 Disparity Results

The disparity map generated by the stereo graph cuts algorithms are all very similar. The only differences are a few pixels along depth discontinuities having different labels. The number of pixels having difficulties in these regions is too small to have a significant effect, as evidenced by the accuracy values discussed in the Section 4.3.2. As a result, we show only one disparity map from for all stereo algorithms (Figure 4.9).

For motion graph cuts algorithms, the disparity maps accuracy varied. Computing motion graph cuts without any multi-resolution implementation resulted in the disparity map in Figure 4.10. The disparity maps for the rest of the motion graph cuts algorithms, those with ID# 6-8 in Table 4.2, are shown in Figure 4.11. Each column represents a level in the pyramid, while each row corresponds to a different motion graph cuts algorithm. The first row is the LS algorithm. The second row is the LDNR algorithm and the third row is the EL algorithm.

The combined stereo-motion disparity maps contain four images, two stereo disparity maps and two motion disparity maps, for each level of the pyramid. This results in sixteen images per algorithm. The LDNR graph cuts method for stereo-motion disparity maps are shown in Figures 4.12 and 4.13. The cause of the black regions in all four disparity maps is a curious behaviour whose cause may be a bug in the code. Figures 4.14 and 4.15 show the disparity maps for the EL graph cuts for stereo-motion algorithm, which are a great improvement over their LDNR stereo-motion algorithm counterparts. Lastly, the most accurate set of disparity maps can be seen in Figures 4.16 and 4.17 for the SAC graph cuts for stereo-motion algorithm.



Figure 4.9: Disparity map for the normal stereo graph cuts algorithm.

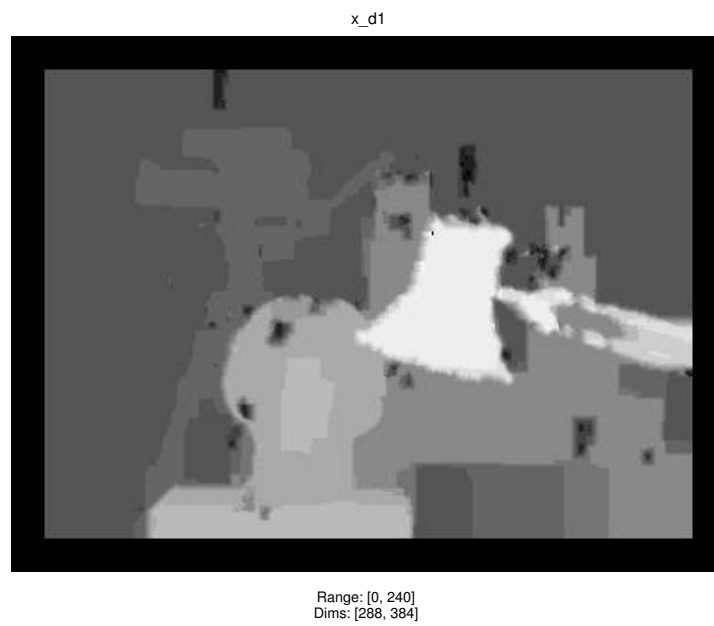


Figure 4.10: Disparity map for the normal motion graph cuts algorithm.

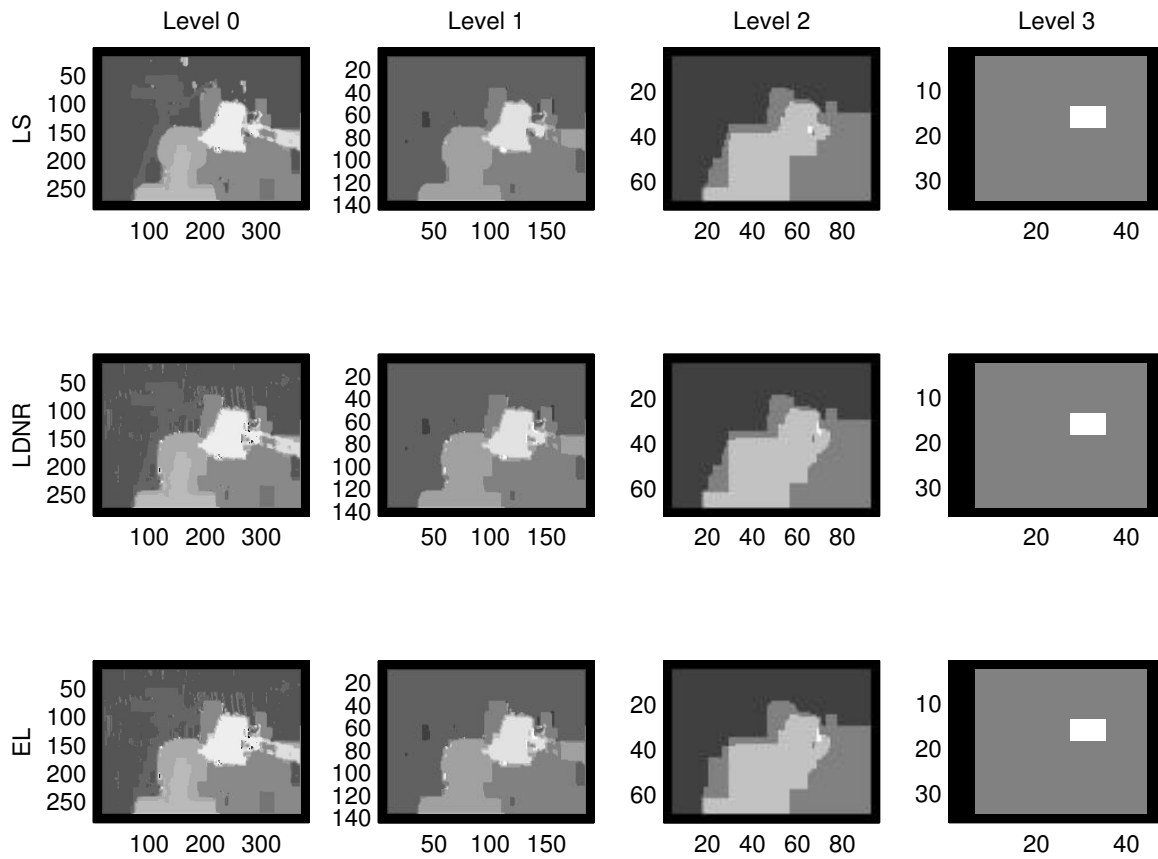


Figure 4.11: Disparity maps for multi-resolution graph cuts for motion algorithms. Each column corresponds to a level in the pyramid operation, while each row corresponds to a different algorithm where the rows are: (1) LS motion graph cuts, (2) LDNR motion graph cuts and (3) EL motion graph cuts.

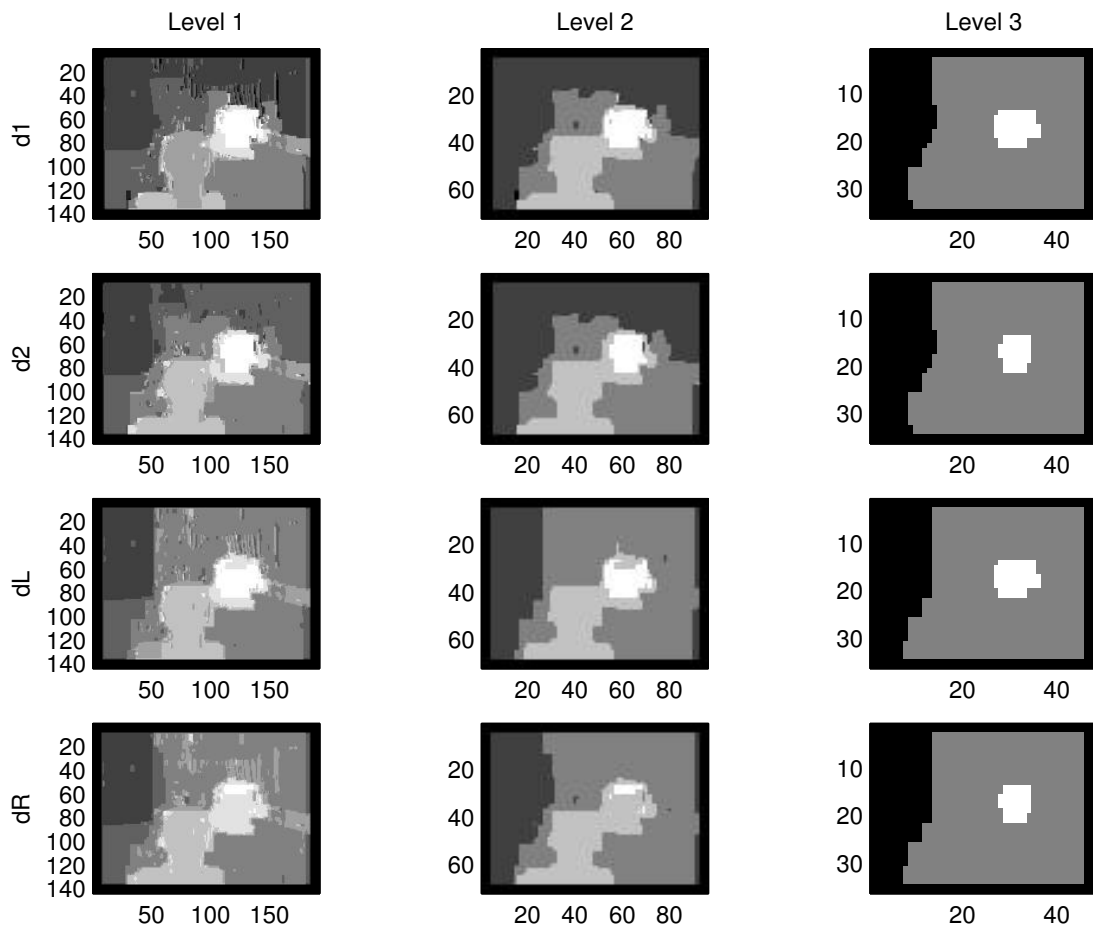


Figure 4.12: Disparity maps for LDNR stereo-motion graph cuts. Each column corresponds to a level of the pyramid, while each row corresponds to the resulting disparity map, with two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R) per level of the pyramid.

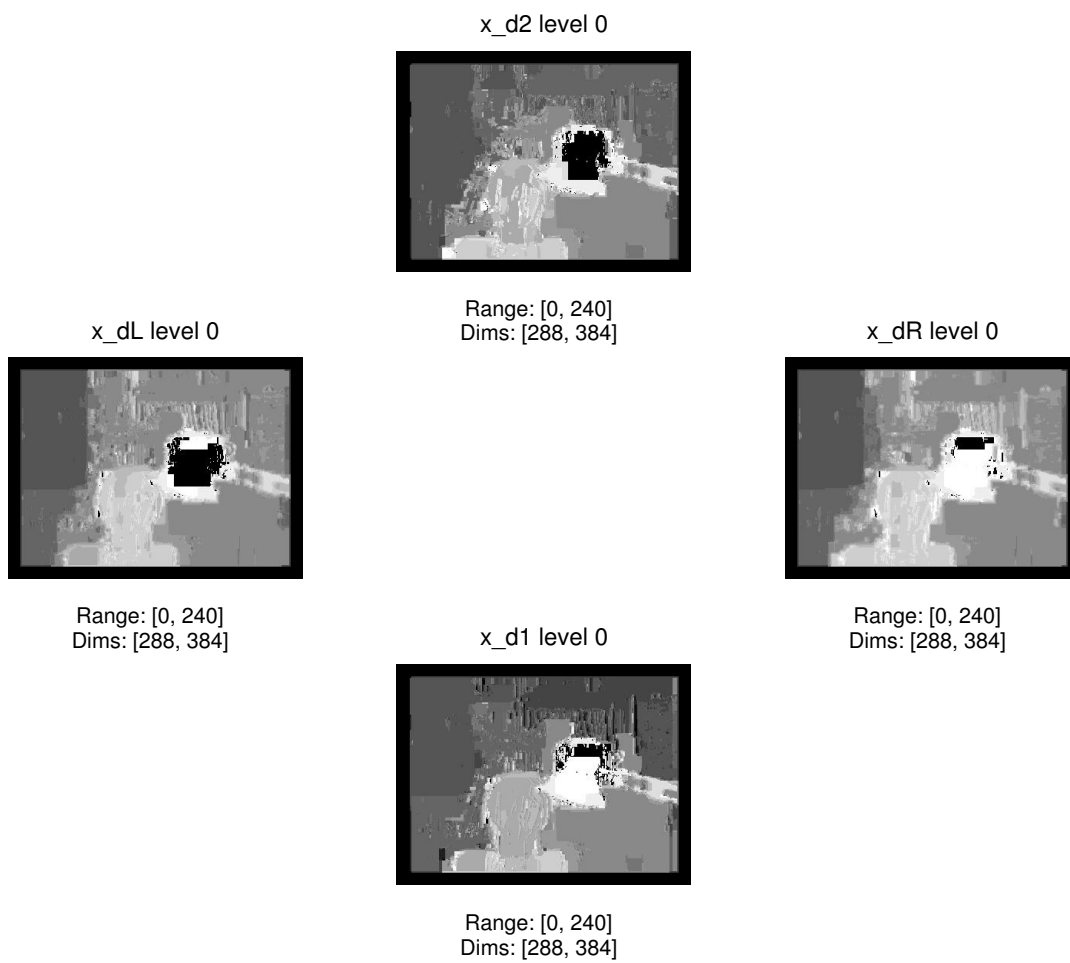


Figure 4.13: Disparity maps for LDNR stereo-motion graph cuts. There are two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R).

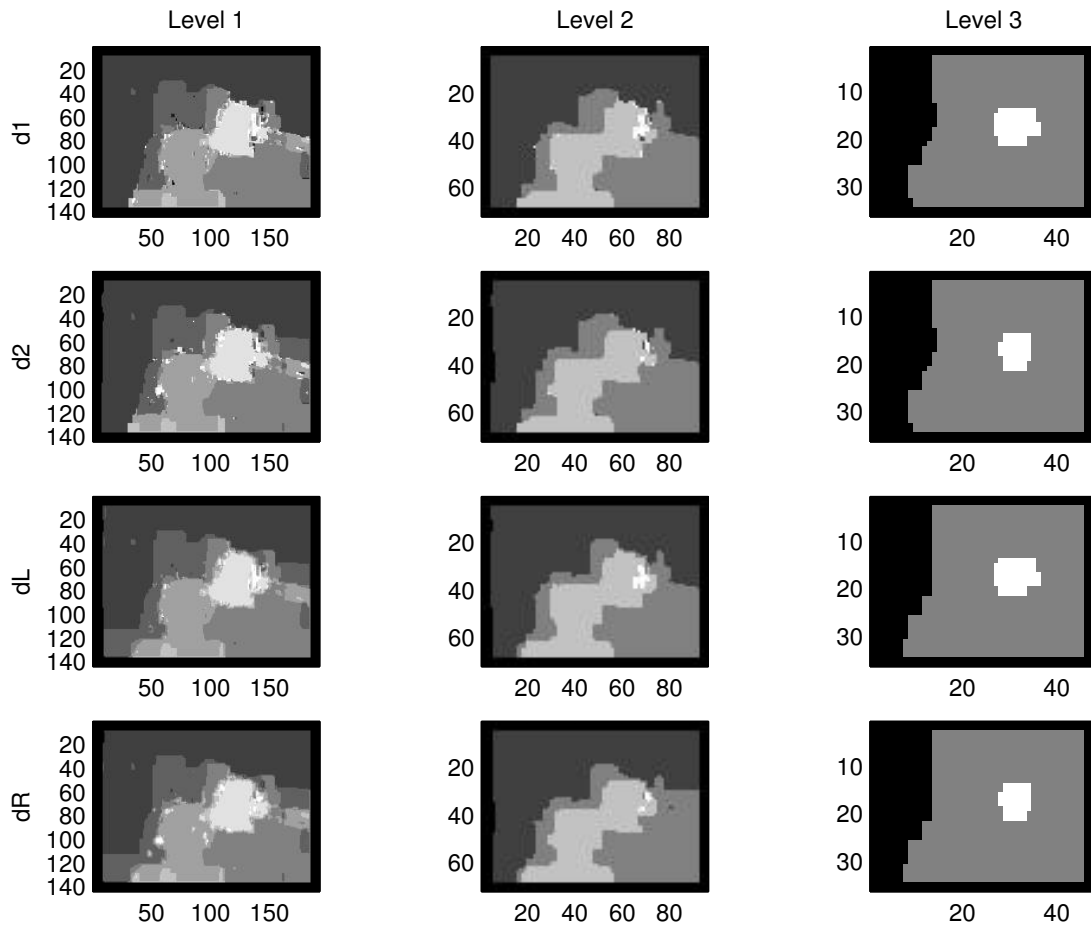


Figure 4.14: Disparity maps for EL stereo-motion graph cuts. Each column corresponds to a level of the pyramid, while each row corresponds to the resulting disparity map, with two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R) per level of the pyramid.

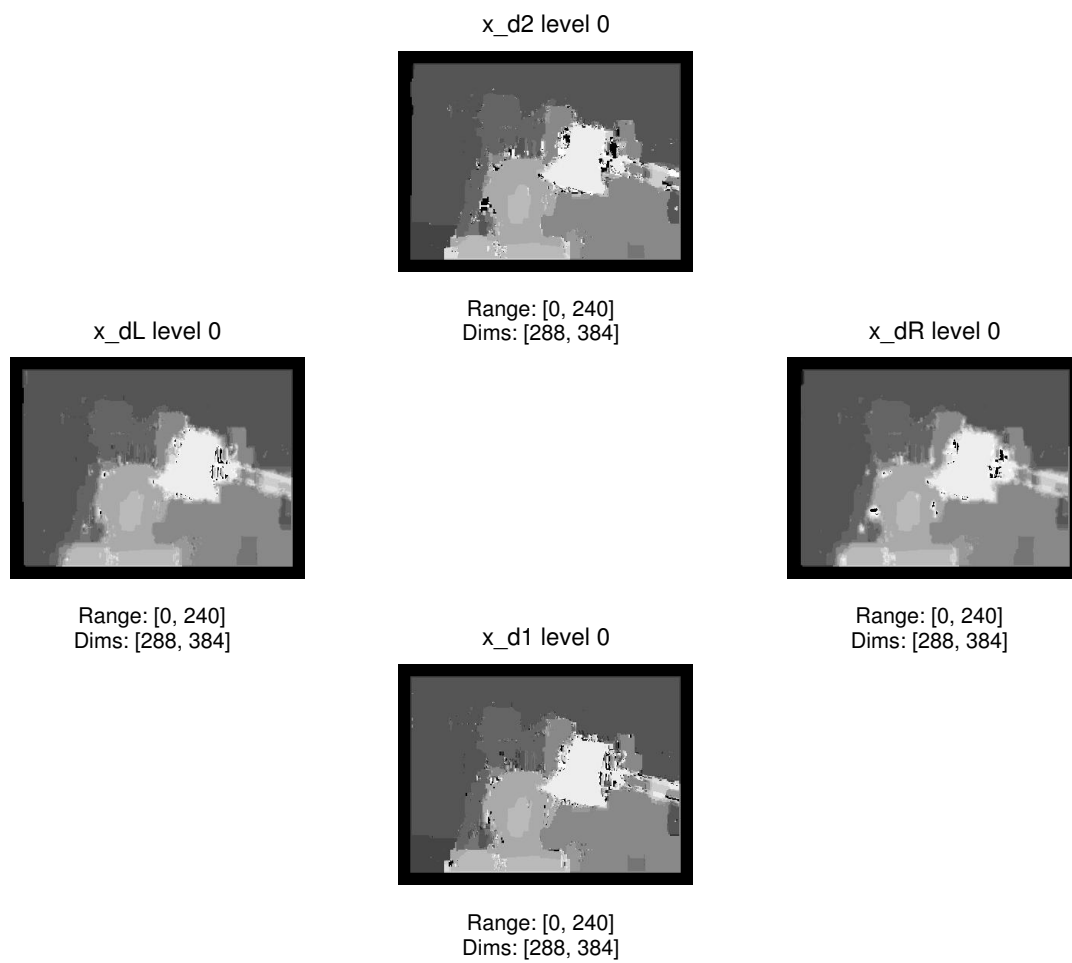


Figure 4.15: Disparity maps for EL stereo-motion graph cuts. There are two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R).

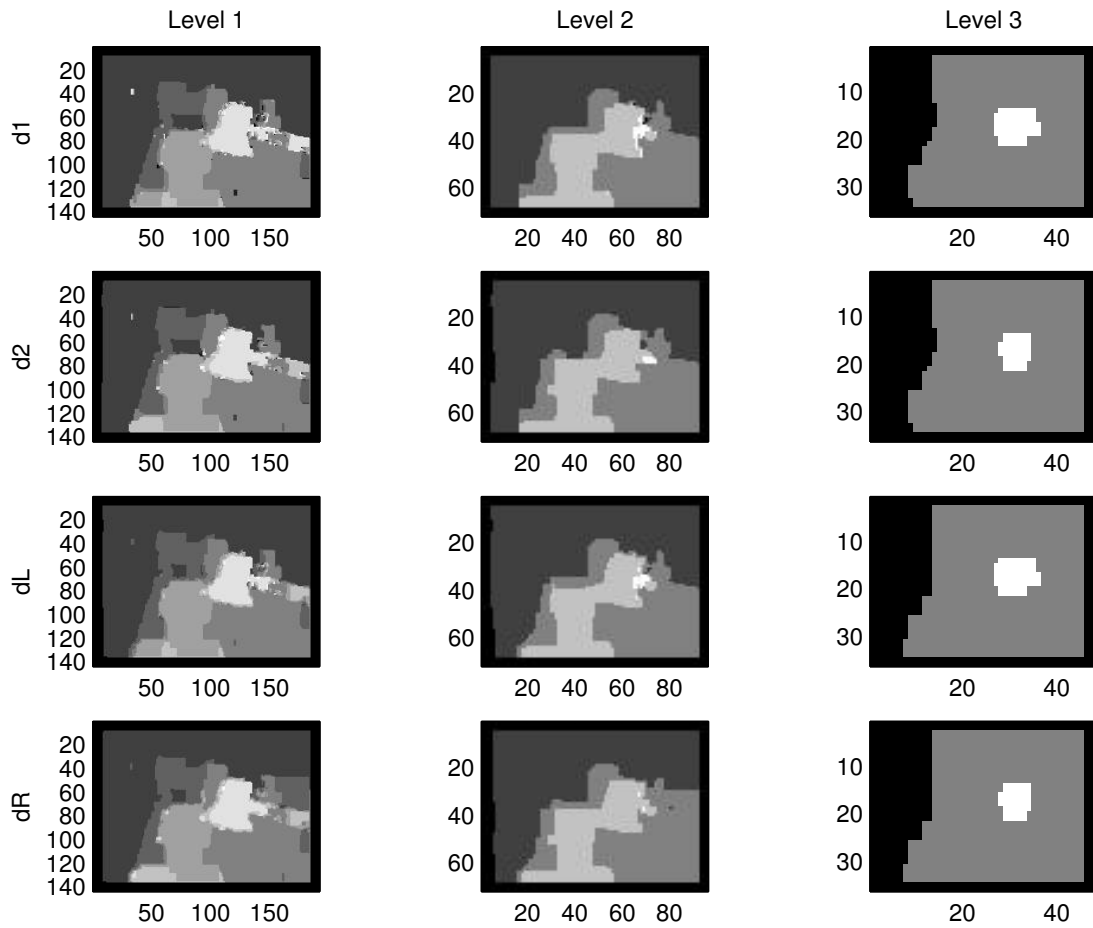


Figure 4.16: Disparity maps for SAC stereo-motion graph cuts. Each column corresponds to a level of the pyramid, while each row corresponds to the resulting disparity map, with two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R) per level of the pyramid.

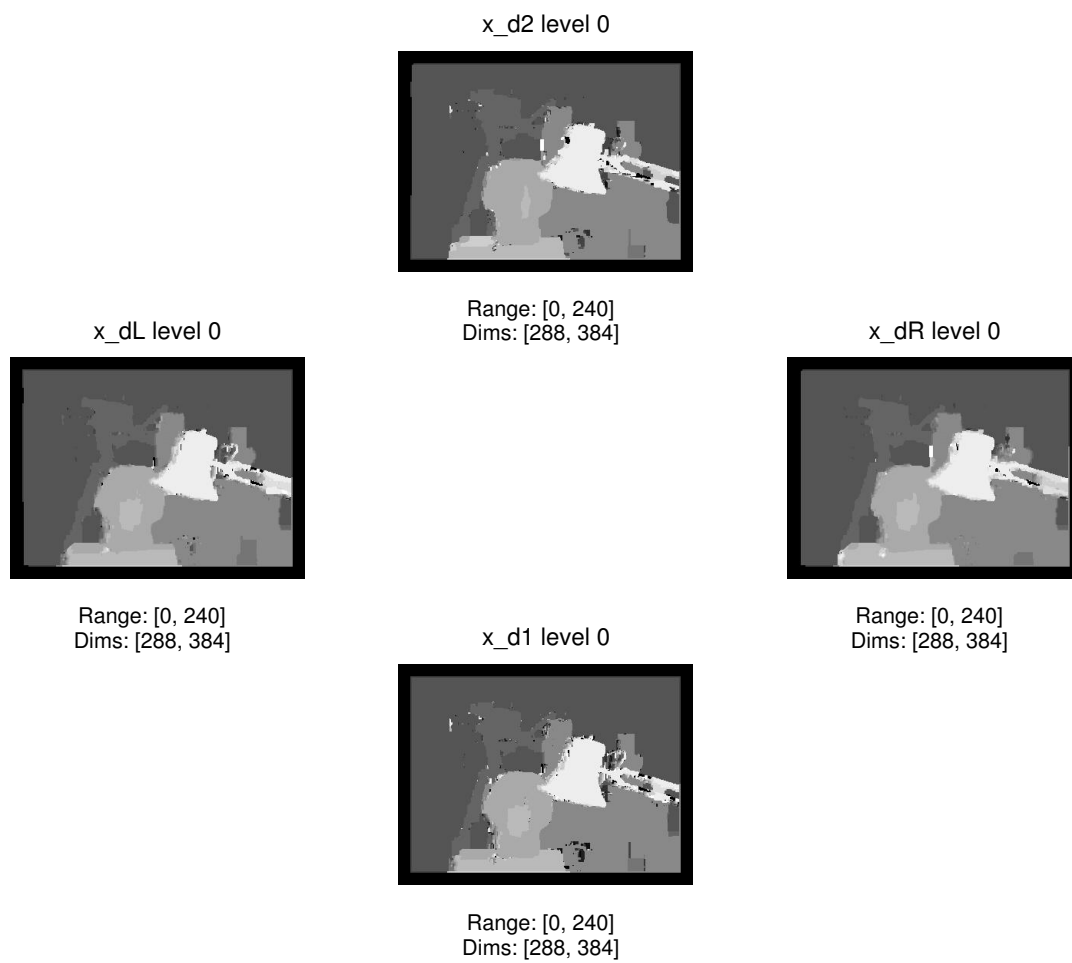


Figure 4.17: Disparity maps for SAC stereo-motion graph cuts. There are two stereo disparity maps (d_1 and d_2) and two motion disparity maps (d_L and d_R).

4.3.2 Analysis

The data in Table 4.3 is illustrated in the accuracy versus time plot of Figure 4.18, which shows all results from algorithms that creates disparity maps for stereo image pairs. These algorithms have the 'ID#' 1-4 and 11-13 in Table 4.2 and Table 4.3, except that those numbered 11-13 have disparity maps for the first stereo pair ($Left_t, Right_t$), labeled as d_1 , and for the second stereo pair ($Left_{t+1}, Right_{t+1}$), labeled as d_2 . In the graph, these stereo pairings become obvious as the data points that are coupled by computation time. For example, the disparity maps d_1 and d_2 for the SAC stereo-motion algorithm have a time value of 533593 seconds.

All stereo algorithms (1-4) implemented achieve similar time and accuracy results with very small variance. Figure 4.20 (left) shows a typical distribution. The error distributions show the absolute difference between the estimated values and the ground truth values in pixels. 97% of the distribution is within one pixel distance from the ground truth data. Those pixels whose error are greater than one pixel usually lie in the regions near depth discontinuities, a common source of error for stereo computation, shown in Figure 4.20 (right) where the intensities are scaled to maximum disparity error.

Looking at the stereo algorithms alone, the benefit of the multi-resolution approach is not apparent. However, if we look at the motion algorithms, the benefits become clear. Figures 4.21 to 4.24 present the distributions (left) and inaccurate label maps (right) for the motion algorithms. As accuracy increases, the computation time increases. The fastest motion algorithm is the LDNR motion algorithm at only 7 seconds computation time, as opposed to the 99 seconds for the normal motion algorithm. This increase in accuracy can reasonably be attributed to the increase in number of label combinations calculated for correspondence.

ID#	Method	Algorithm	Accuracy%	RMSE(pixs)	Time (s)
1	Stereo	Normal	97.406	0.912	23
2	Stereo	LS	97.399	0.914	21
3	Stereo	LDNR	97.402	0.901	19
4	Stereo	EL	97.399	0.914	20
11	SM	LDNR (d1)	80.71	2.200	294
		LDNR (d2)	69.42	3.230	294
12	SM	EL (d1)	87.18	1.680	2073
		EL (d2)	86.37	1.790	2073
13	SM	SAC (d1)	90.96	1.340	533593
		SAC (d2)	91.12	1.350	533593

Table 4.3: The relationship between system accuracy and time to convergence for graph cuts algorithms creating stereo disparity maps, where SM represents stereo-motion.

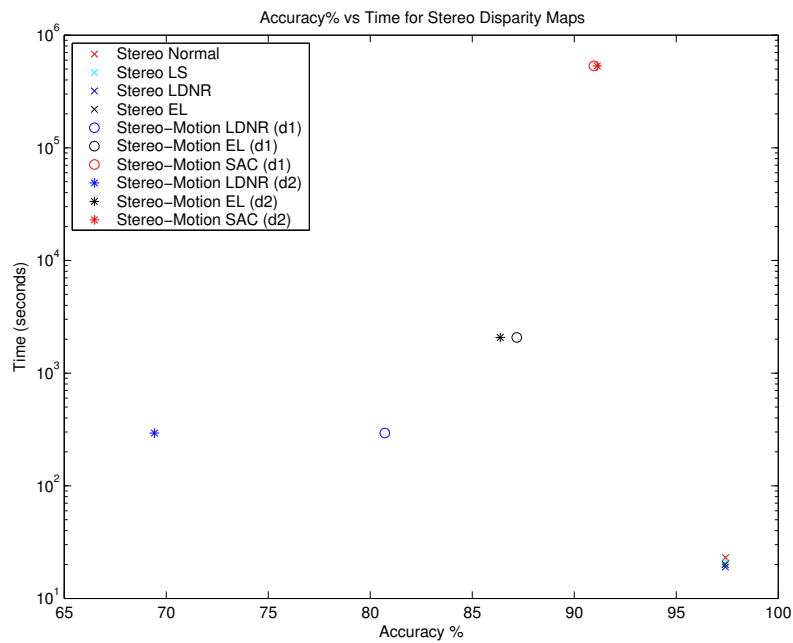


Figure 4.18: The accuracy versus time plots for all algorithms that create stereo disparity maps (d1 and d2).

ID#	Method	Algorithm	Accuracy%	RMSE(pixs)	Time (s)
5	Motion	Normal	93.671	1.108	99
6	Motion	LS	90.168	1.267	48
7	Motion	LDNR	84.577	1.412	7
8	Motion	EL	87.329	1.438	11
11	SM	LDNR (dL)	46.75	3.53	294
		LDNR (dR)	46.59	3.14	294
12	SM	EL (dL)	86.19	1.54	2073
		EL (dR)	84.66	1.70	2073
13	SM	SAC (dL)	91.50	1.25	533593
		SAC (dR)	90.55	1.32	533593

Table 4.4: The relationship between system accuracy and time to convergence for graph cuts algorithms creating motion disparity maps, where SM represents stereo-motion.

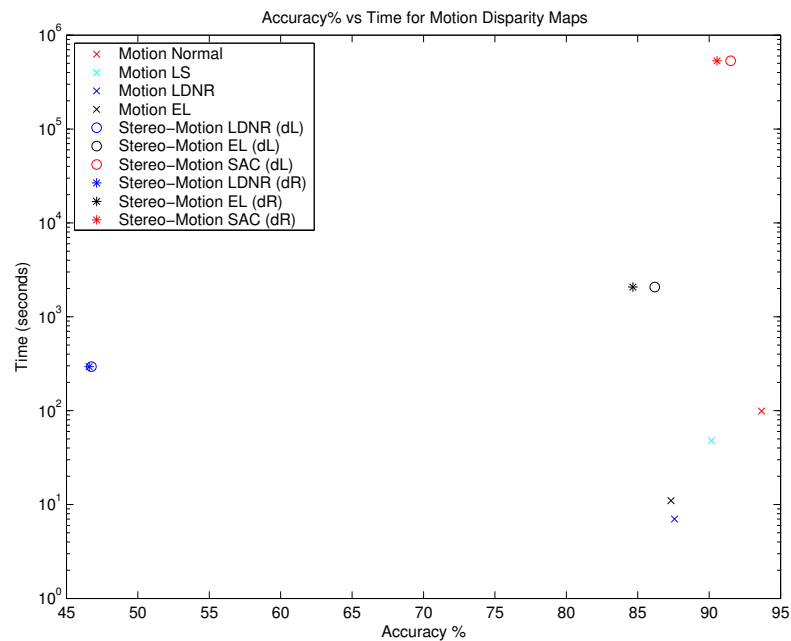


Figure 4.19: The accuracy versus time plots for all algorithms that create motion disparity maps (dL and dR).

Examining the combined stereo-motion algorithms further reinforces this relationship. Recall from Section 3.8, that each of the combined stereo-motion algorithms limits the label set in different ways. The result is that the LDNR algorithm severely reduces the number of label combinations, from 83521 to 251 labels, thereby significantly reducing the computation time, from an estimated 6 months computation time down to 294 seconds. If we allowed the normal stereo-motion graph cuts algorithm to run for this same amount of time, it would barely finish a fraction of the computation needed to complete the first iteration. However, as previously mentioned, accuracy suffers greatly, reducing down to a 70-80% accuracy rate. This is illustrated in the error distributions and inaccurate label maps of the stereo-motion graph cuts algorithms in Figures 4.25 to 4.30. As we continually increase the number of labels and consequently the computation time, as in EL stereo-motion graph cuts and SAC stereo-motion graph cuts, we improve the system accuracy. We can attribute this improved accuracy to solving the error propagation problem, described in (Section 3.8.2), for pixels near depth discontinuities and the background layer. These pixels are allowed the opportunity to take on similar labels to the pixels in these regions that are already labeled properly. This is evidenced in the stereo-motion inaccurate label maps in Figure 4.26 for the LDNR algorithm, Figure 4.28 for the EL algorithm and Figure 4.30 for the SAC algorithm, where a large portion of the improperly labeled pixels in the background and depth discontinuity regions of the LDNR algorithm disappear in the EL algorithm and almost completely disappear in the SAC algorithm.

The benefit of the multi-resolution approach is more pronounced if we look at both the motion graph cuts algorithms and the combined stereo-motion graph cuts algorithms. Table 4.4 and its graphical representation in Figure 4.19 demonstrate this difference between accuracy and computation time of the resulting motion disparity map estimates. The motion disparity map estimates are generated in algorithms that have the ID# 5-8 and 11-13 in Table 4.2 and Table 4.4, with the exception that those numbered 11-13

have disparity maps for the left motion pair $(Left_t, Left_{t+1})$, labeled as d_L , and the right motion pair $(Right_t, Right_{t+1})$, labeled as d_R . Using any of the different multi-resolution algorithms for motion computation had a large effect in reducing computation time down to a fraction of the original motion graph cuts algorithm; from 98 seconds down to 7 seconds in the case of the LDNR algorithm. There was only a slight loss in accuracy.

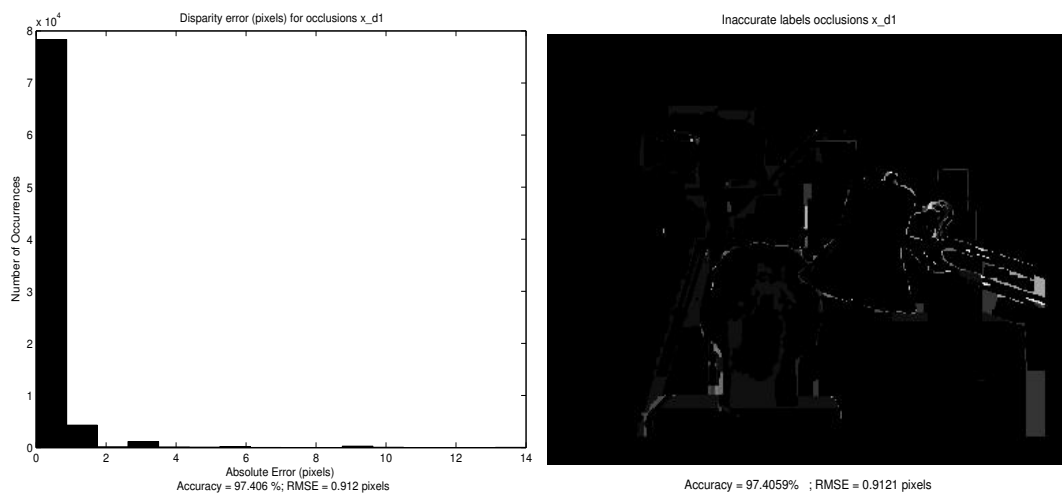


Figure 4.20: Distribution of pixel error (left) and inaccurate label map (right) for the original stereo graph cut algorithm.

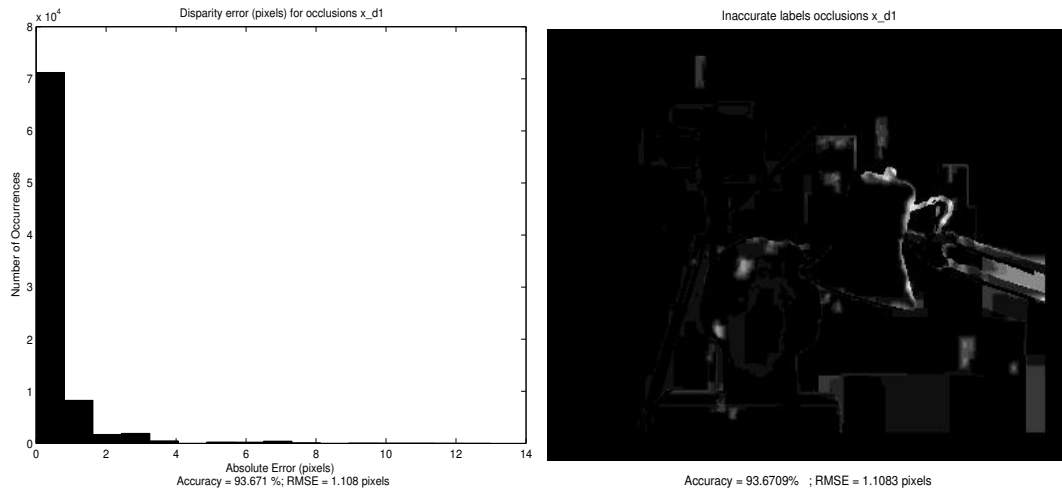


Figure 4.21: Distribution of pixel error (left) and inaccurate label map (right) for the motion graph cut algorithms without any multi-resolution implementation.

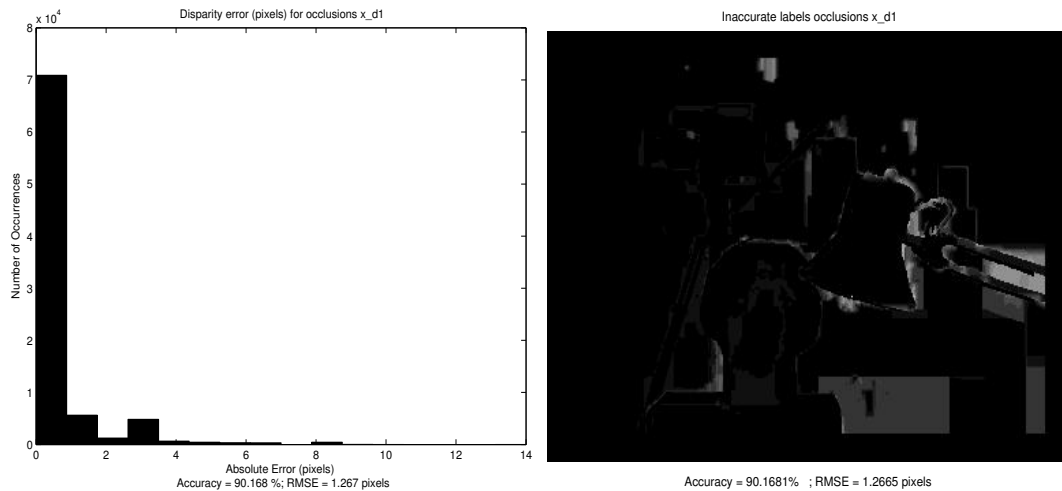


Figure 4.22: Distribution of pixel error (left) and inaccurate label map (right) for the LS motion graph cut algorithms without any multi-resolution implementation.

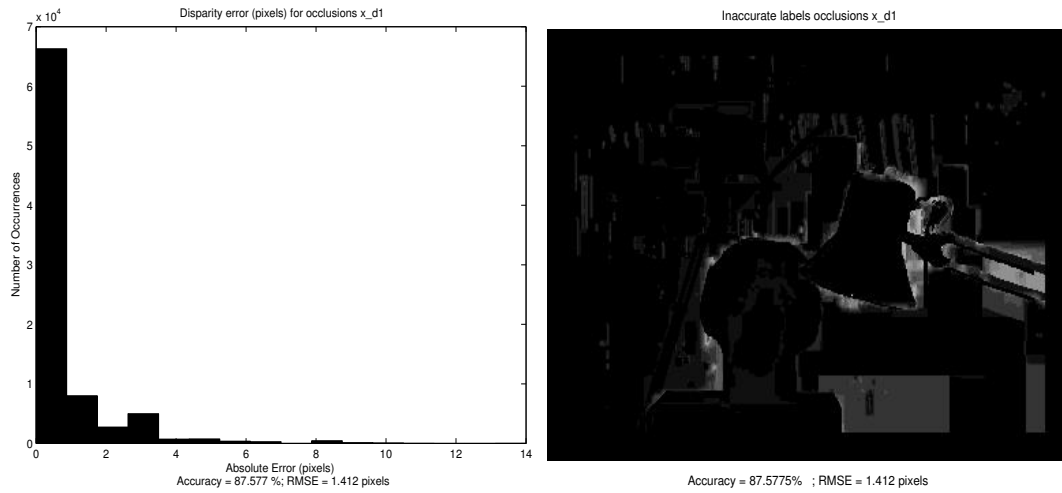


Figure 4.23: Distribution of pixel error (left) and inaccurate label map (right) for the LDNR motion graph cut algorithms without any multi-resolution implementation.

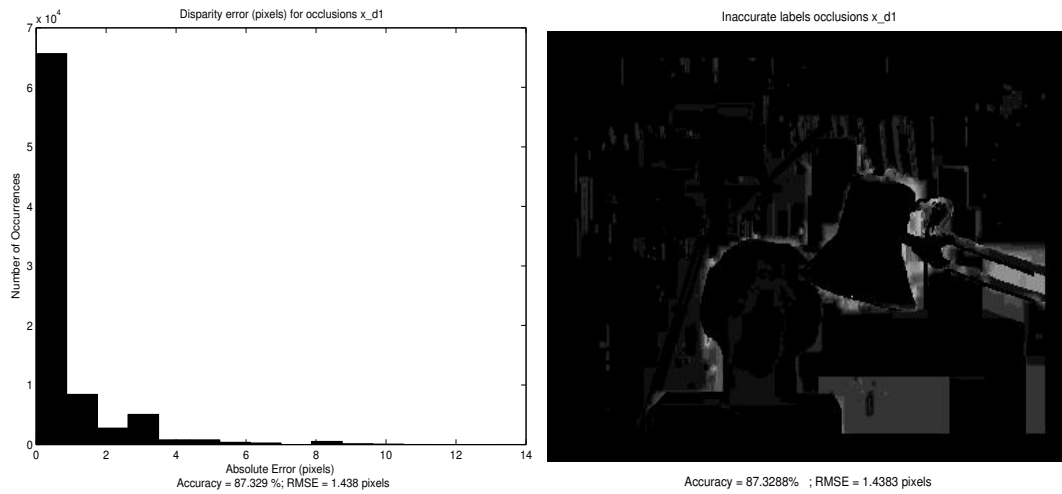


Figure 4.24: Distribution of pixel error (left) and inaccurate label map (right) for the EL motion graph cut algorithms without any multi-resolution implementation.

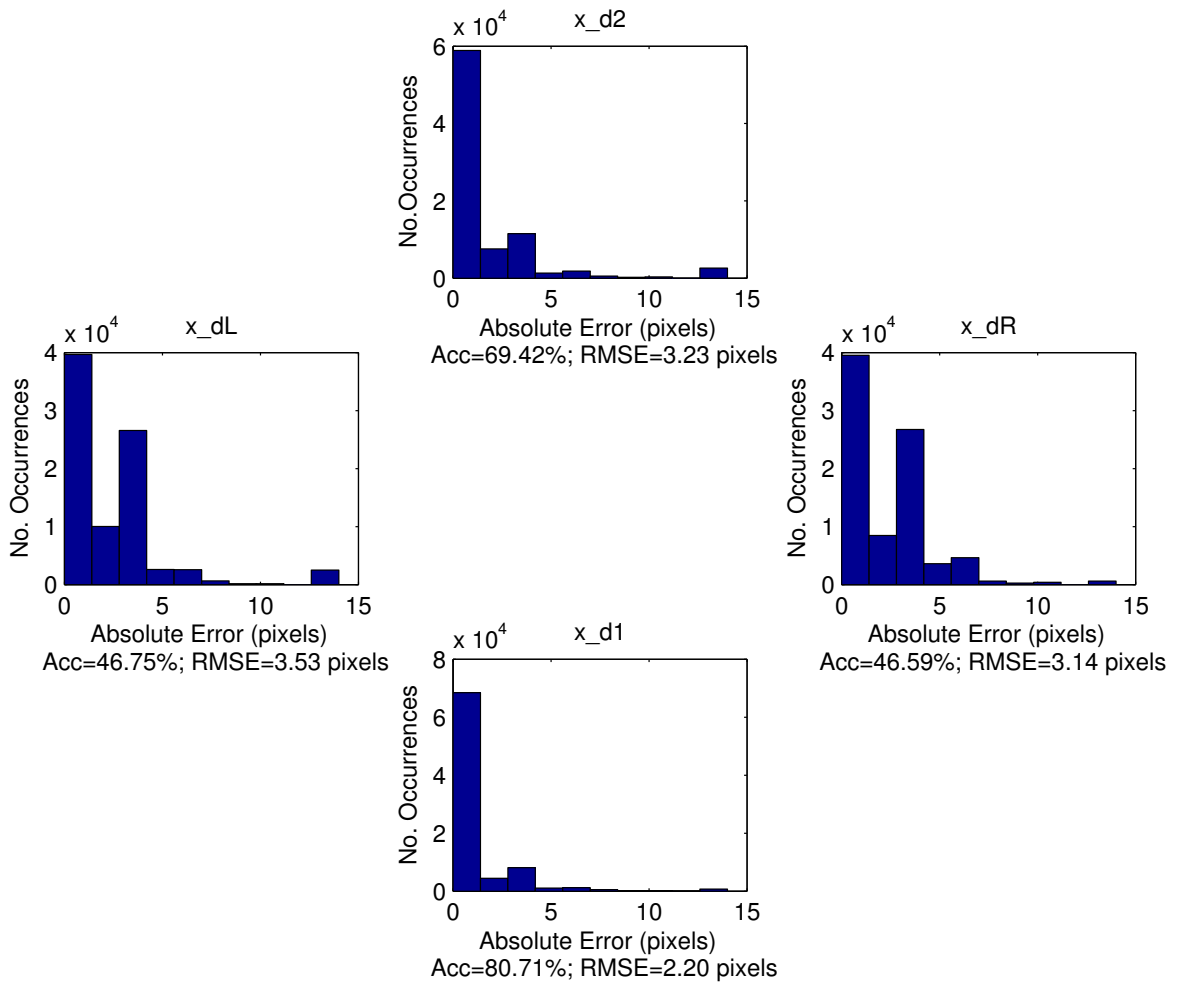


Figure 4.25: Distribution of pixel error for LDNR stereo-motion graph cut algorithms.

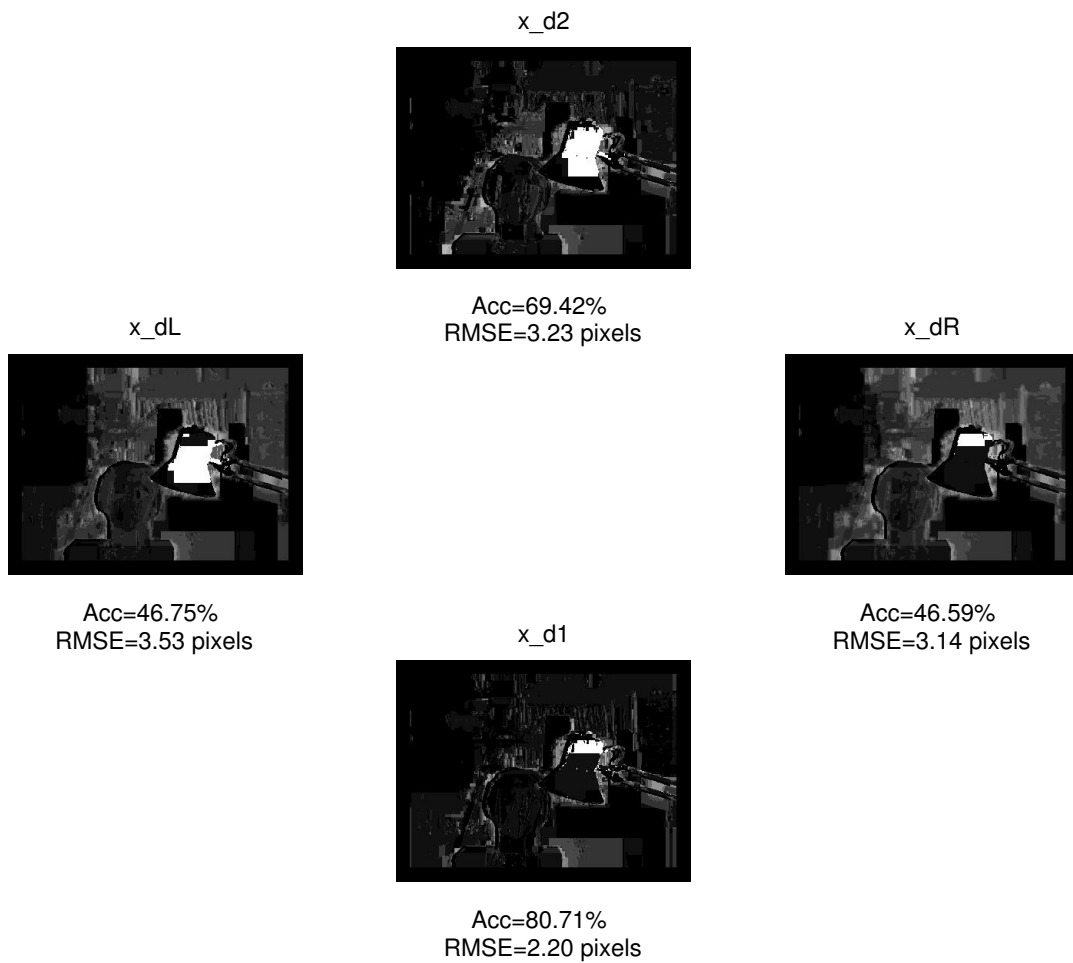


Figure 4.26: Inaccurate label map for LDNR stereo-motion graph cut algorithms. Brighter pixels indicate greater error.

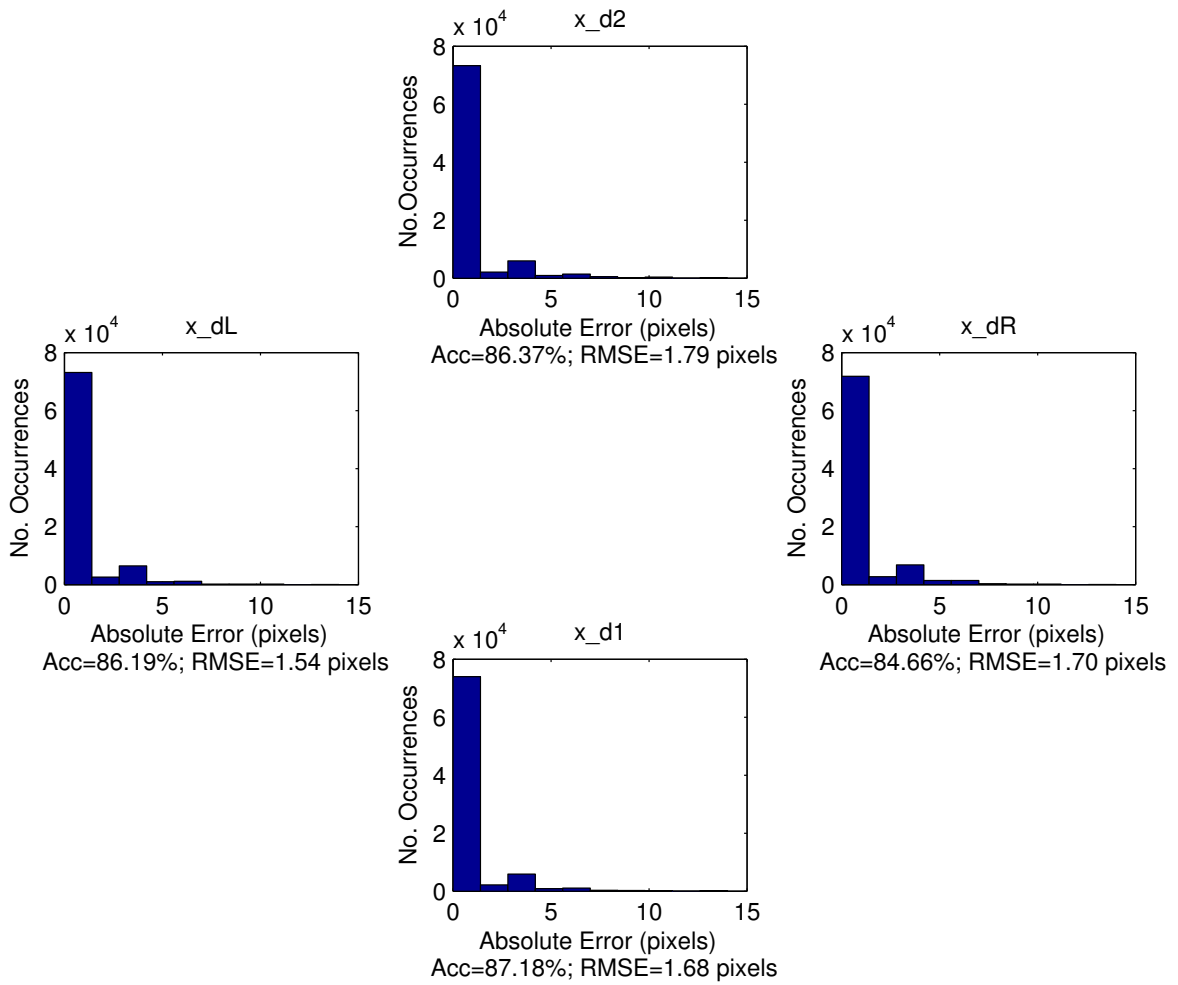


Figure 4.27: Distribution of pixel error for EL stereo-motion graph cut algorithms.

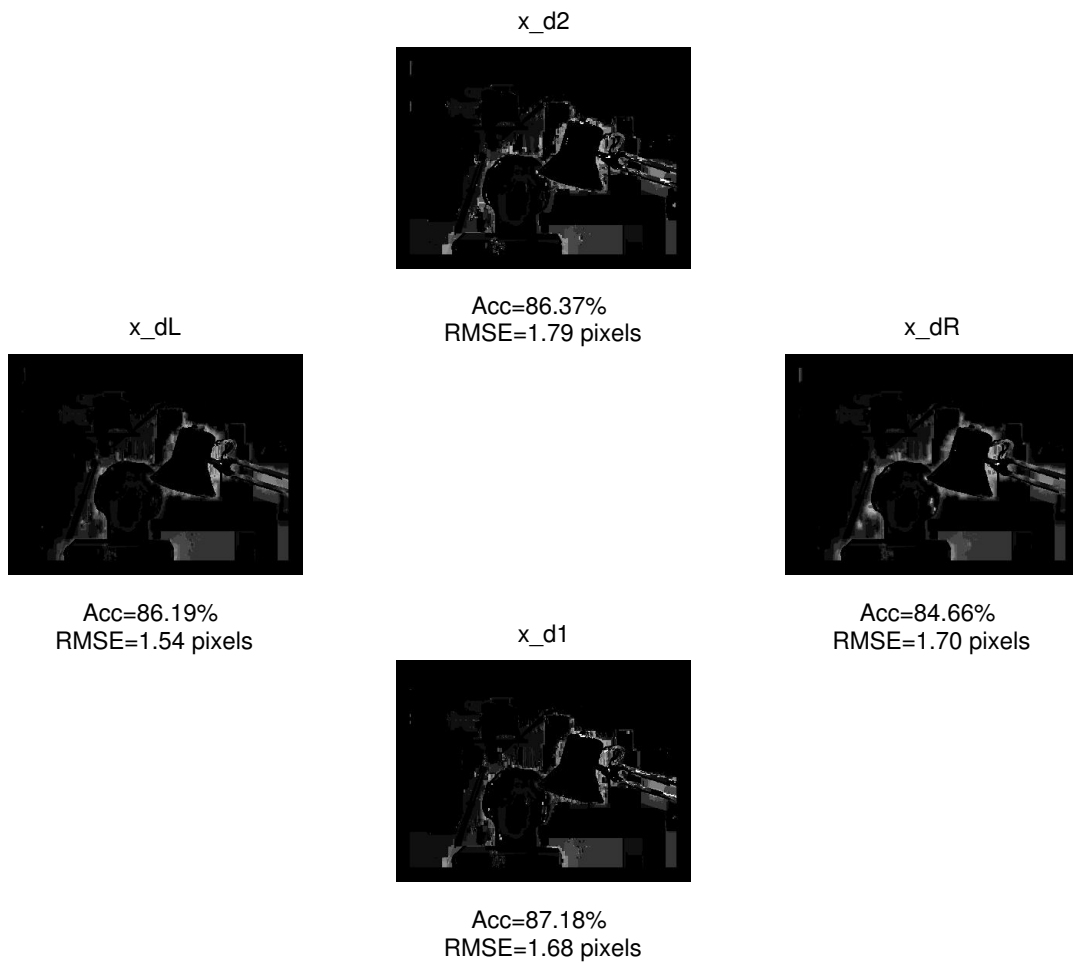


Figure 4.28: Inaccurate label map for EL stereo-motion graph cut algorithms. Brighter pixels indicate greater error.

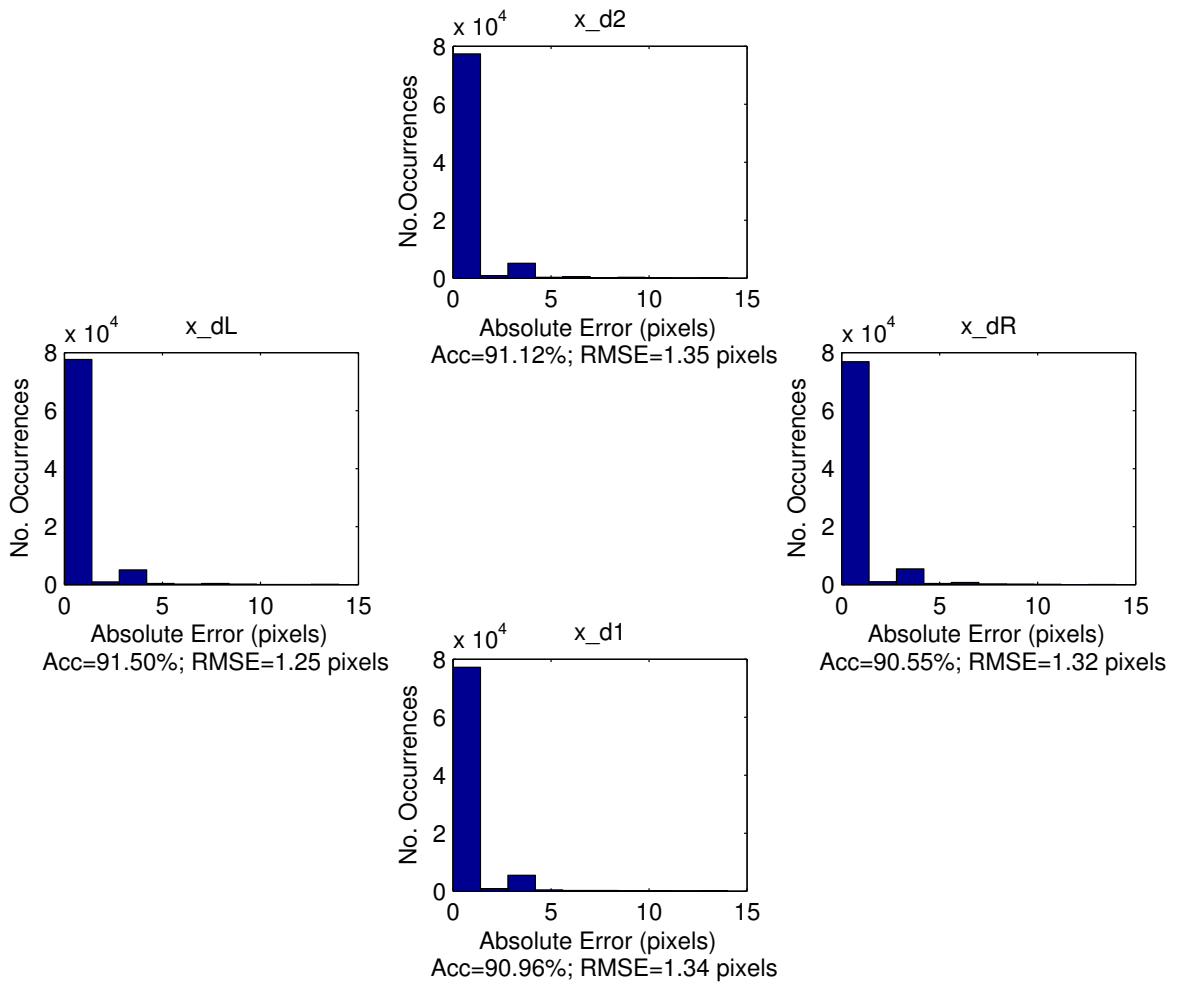


Figure 4.29: Distribution of pixel error for SAC stereo-motion graph cut algorithms.

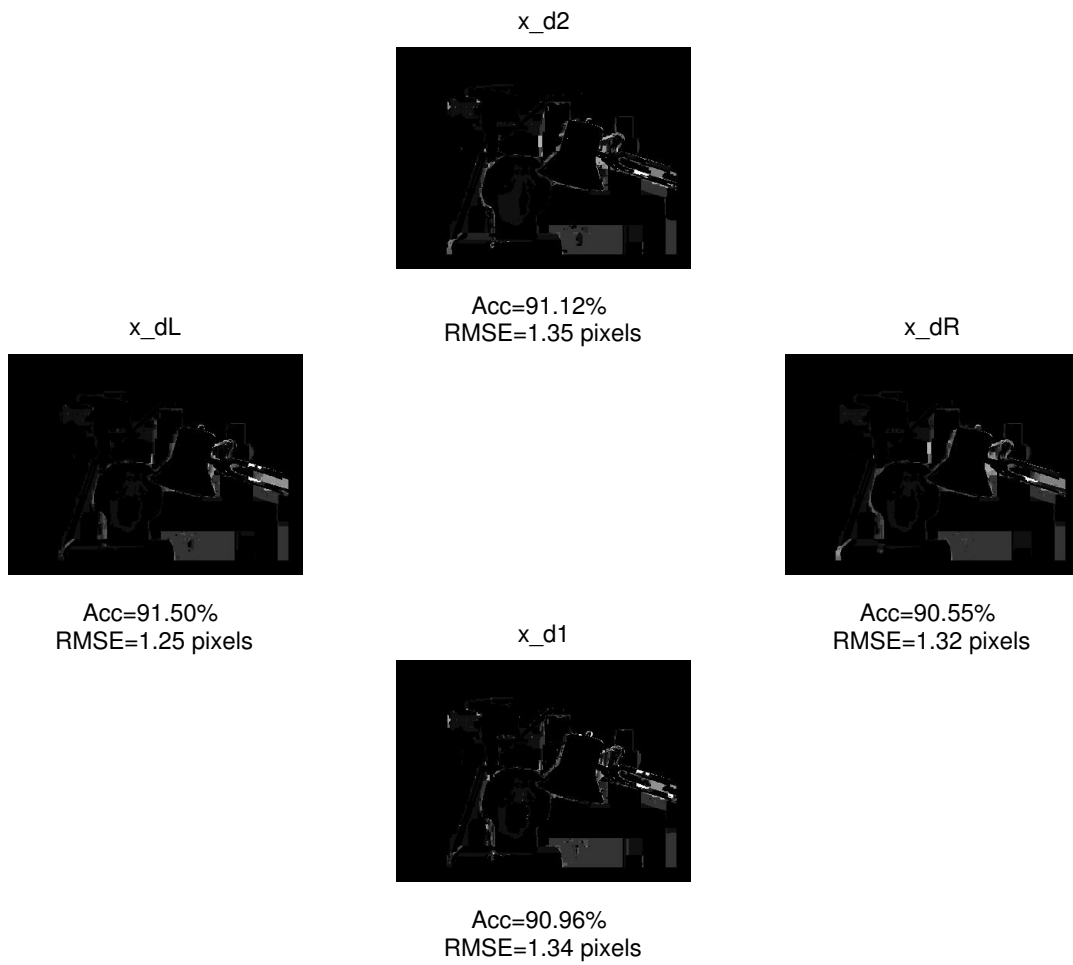


Figure 4.30: Inaccurate label map for SAC stereo-motion graph cut algorithms. Brighter pixels indicate greater error.

ID#	Method	Algorithm	Energy	Time (secs)
11	SM	LDNR (dL)	7652980	294
12	SM	EL (dL)	1914927	2073
13	SM	SAC (dL)	717592	533593

Table 4.5: The resulting energy values for each stereo-motion algorithm.

We can also use the final energy value of the resulting energy functional as a measure of goodness of a solution. The lower the energy value, the closer the solution is to the global minimum. Figure 4.31 is an example of the total system energy over the course of its operation. Two of the lines in each of the graphs indicate the energy value of the components that make up the energy function, namely the data energy term E_{data} and the smoothness energy term E_{smooth} . The third line represents the total energy of the system, which is the sum of the data and smoothness terms. The vertical dotted lines have the same representation as those in the number of labels versus time graphs of Section 4.2, where each line represents a change in level of the pyramid. Thus, the large spikes in energy coincide with each change in level of computation because of the increase in pixels and labels that cause larger energy values when computing both the data and smoothness terms.

All three of these plots represent the energy value over time for three stereo-motion graph cuts algorithms. Figure 4.31 (a) corresponds to the LDNR algorithm. Figure 4.31 (b) shows the EL algorithm energy results, while Figure 4.31 (c) gives the SAC algorithm energy results. The final energy values achieved by each algorithm are summarized in Table 4.5. The SAC algorithm achieves a lower energy measure, and as such, is more accurate than the other two stereo-motion methods.

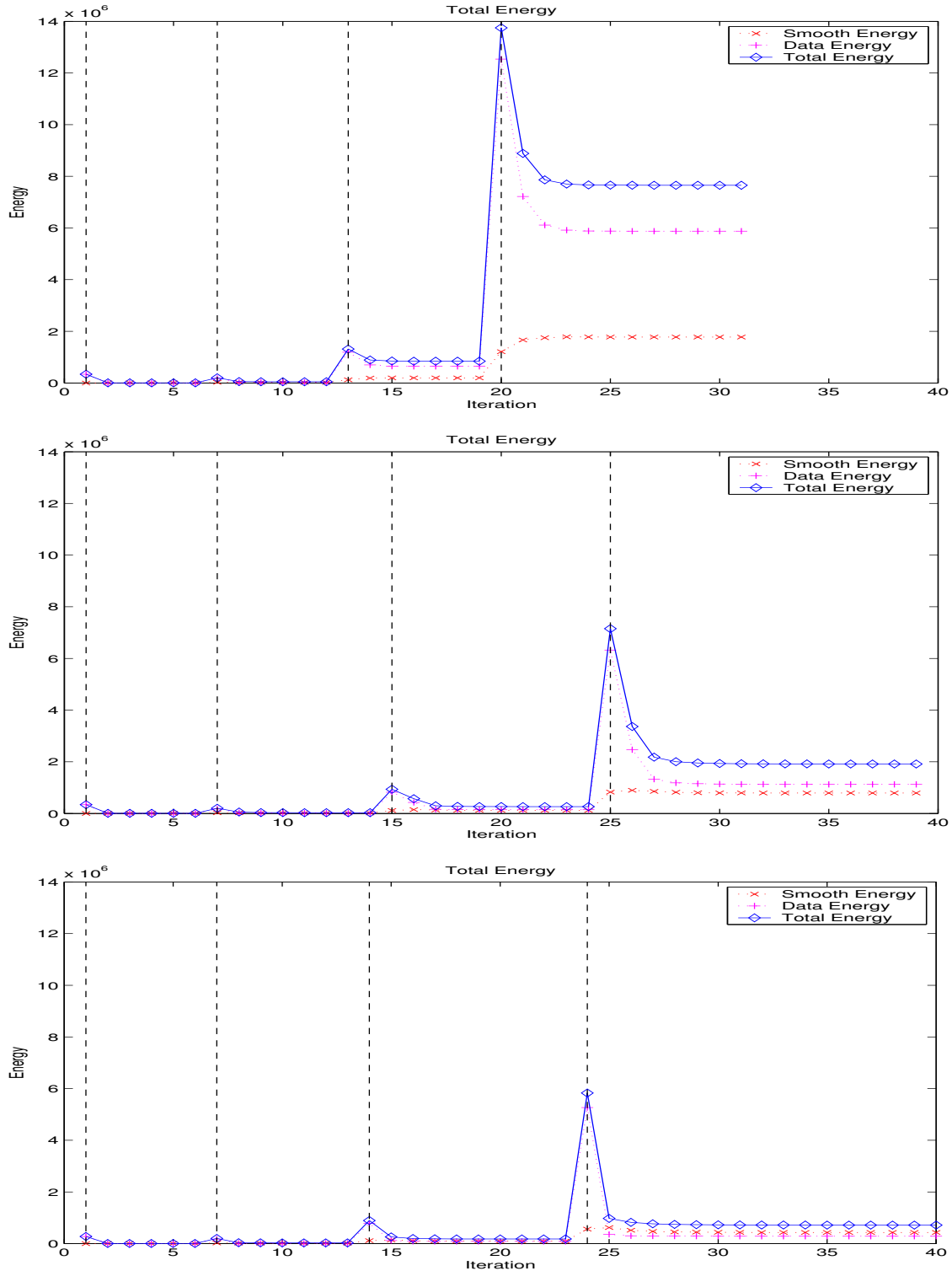


Figure 4.31: Total energy graph for the stereo-motion graph cuts algorithms: (top) LDNR algorithm, (middle) EL algorithm, and (bottom) SAC algorithm. Vertical dotted lines indicate a change in level operation in the MRGC algorithm. The first dotted line on the left begins at the highest (coarsest) level of the pyramid operation, finishing with the lowest (finest) level of the pyramid. This example begins at level 3 and finishes at level 0.

4.4 Summary

In Chapter 4, we began with a demonstration of the image pyramids created by the system. Next, we looked at the effect of varying the number of labels on the system, illustrating that the system computation time has a direct relationship to the number of pixels. Increasing the number of labels increases the computation time, independent of the choice of graph cuts algorithm. However, there is a linear relationship for graph cuts implementing the α -expansion algorithm as opposed to a quadratic relationship for graph cuts implementing α - β swap algorithm.

System accuracy also had a quadratic relationship to computation time in the multi-resolution graph cuts algorithms for stereo-motion. Each of the three algorithms implemented reduced the label sets, but at different degrees, directly affecting system accuracy. Also affecting accuracy are regions of depth discontinuities and occlusion.

In the next chapter, we suggest ways to improve the computation time of the system. These include the use of the fundamental matrix, determining a different convergence criteria and formulating a metric smoothness function. We also propose a method to compute sub-pixel disparity measurements for graph cuts.

Chapter 5

Conclusions and Future Work

The original graph cuts algorithm, presented in [6, 7, 8, 5, 18, 28], produces very accurate disparity maps at the expense of high computation cost. The method includes two algorithms used for graph creation, the α -expansion algorithm and the α - β swap algorithm, with the decision factor being the formulation of the smoothness term. If the designed smoothness function is a metric, one should use the α -expansion algorithm, with a time complexity of $O(mn)$, where m is the number of pixels and n is the number of labels. This algorithm has the benefit of a guaranteed solution within a known factor of 2 of the global minimum. However, formulating a metric function creates difficulties for more general problems, such as motion. In these cases, it is easier to create a semi-metric smoothness function, requiring the use of the much slower α - β swap algorithm, which is $O(mn^2)$. Consequently, the influence of the label set size becomes even more pronounced.

Taking this into consideration, this thesis presents a multi-resolution graph cuts for a stereo-motion system. The multi-resolution feature of the system design creates a framework with many benefits. First, we are able to handle larger images than the original graph cuts algorithm. The multi-resolution framework breaks the correspondence problem into smaller, faster subproblems of computing correspondence on smaller images with smaller label sets. This allows for an initialisation of the subsequent pyramid level

to be closer to that level's objective function's minimum. By the time we reach the finest level of the pyramid, we are already very close to the best solution that the α - β swap algorithm can provide. As a result, the number of iterations to reach convergence, and conversely computation time, is decreased. Second, we are able to handle larger disparity ranges. In a similar fashion to the handling of larger images, the multi-resolution framework also reduces the label set size at each level of the pyramid. Third, this multi-resolution framework helps promote solutions to more general problems, such as motion estimation, where the search space lies in two-dimensions, as opposed to one dimension for rectified stereo.

Another system design feature is the use of both stereo and motion constraints to improve the accuracy. The resulting effect on accuracy is unknown at the moment, but is still under computation because of the increased time complexity in the problem. Using this stereo-motion framework, the problem extended from a one-dimensional problem in stereo to a five-dimensional problem, one-dimension for each of the two stereo image pairs and two-dimensions for the motion image pairs (two horizontal and one vertical due to image rectification). The result of this extension requires the use of the slower α - β swap algorithm for graph creation, thus increasing computation time. However, in this framework, we are still able to establish the trade-off between an increase in computation time that will result in an increase in accuracy.

A highlight of the system is the design of three different approaches to multi-resolution graph cuts. The first algorithm, LDNR, uses a vastly reduced set of labels to compute the resulting disparity maps. It is the fastest of the three algorithms and is best used in situations where fast initialisation close to the global minimum is desired, because the overall accuracy rates suffers. However, for a substantial increase in accuracy (if an increase in computation time is acceptable) the choice of the EL algorithm is a better option. Lastly, if accuracy is of greatest importance, the use of the SAC algorithm is the best option. The label sets are less than straightforward computation, due to the

multi-resolution framework, but are significant enough to increase accuracy by 5-6% when used with stereo-motion computation. There is, however, a big increase in computation time. On the other hand, when compared to the straightforward computation of stereo-motion with a multi-resolution approach, there is a significant decrease in computation time, going from an estimated computation time of 6 months to 5-6 days (depending on convergence limits).

Output of the system is in the form of dense disparity maps. The benefit of dense disparity maps lies in their inverse relationship with depth. Given that we are able to obtain dense disparity maps, we are able to discern the depth to objects in the scene. This becomes very useful in applications such as scene reconstruction, navigation, image segmentation, object recognition and object tracking.

5.1 Future Work

The multi-resolution framework described in this thesis provides a general framework that opens up the possibility of many areas of future work. Areas for improvement involve improving both the accuracy and the computation time of the system. Some of these areas are:

- Finding the fundamental matrix between motion image pairs, according to [14], so that the search space for correspondence between becomes one-dimensional. From the fundamental matrix, we know that a point in one image must have a corresponding point lie along the epipolar line in the other image, assuming no independently moving objects in the scene. Therefore, we have reduced the overall system search space from five dimensions (two for stereo and three for motion) to four dimensions (two for stereo and two for motion), thereby improving computation time and accuracy. This is valid if there are no independently moving objects in the images.

- Formulate a smoothness term in the form of a metric, thus sacrificing accuracy for a significant reduction in computation time. Using a metric would allow the use of the α -expansion algorithm during the graph building step. The α -expansion algorithm is $O(mn)$ as opposed to $O(mn^2)$ in the α - β swap algorithm. It also has the guarantee of finding a solution within a known factor of 2 from the global minimum.
- Determine an improved convergence criteria for the energy minimisation technique. Generally, after 4-5 iterations, the system has nearly stabilized, with its energy being very close to the energy of the system it would achieve at convergence. This signifies that the solutions are quite similar and implies that changes that are occurring are labeling changes of only a few pixels at a time. The pixels are most likely those pixels in the regions of difficulty: occlusion, textureless or near depth discontinuities. Improving the convergence criteria can help improve system speed with only a little decrease in accuracy.
- Compute sub-pixel accuracy disparity measurements. The original graph cuts algorithm is only setup for integer values of disparity. However, in the multi-resolution framework created in this thesis, we are able to handle larger label sets. A good approach would be to initially calculate disparities to integer accuracy and then sub-divide each integer value into smaller floating point labels and computing correspondence in each reduced set of labels.

References

- [1] S.S. Beauchemin and J.L. Barron. The computation of optic flow. *ACM Computing Surveys*, 27:433–467, September 1995.
- [2] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.
- [3] M.J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *European Conference on Computer Vision*, pages 231–236, 1993.
- [4] A.F. Bobick and S.S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.
- [5] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.
- [6] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *IEEE Computer Society Conference on Computer Vision & Pattern Recognition*, page 648, 1998.
- [7] Y. Boykov, O. Veksler, and R. Zabih. A new algorithm for energy minimization with discontinuities. *IEEE Computer Society Conference on Computer Vision & Pattern Recognition*, pages 205–220, July 1999.

- [8] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.
- [9] D. Cheung. Motion segmentation incorporating active contours for spatial coherence. Master’s thesis, University of Toronto, 2004.
- [10] P.B. Chou and C.M. Brown. The theory and practice of bayesian image labeling. In *International Journal of Computer Vision*, volume 4, pages 185–210, 1990.
- [11] L. Ford and D. Fulkerson. *Flow in Networks*. Princeton University Press, 1962.
- [12] D. Geiger and F. Girosi. Mean field theory for surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):401–412, 1991.
- [13] 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.
- [14] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2nd edition, 2003.
- [15] A.Y.K. Ho and T.C. Pong. Cooperative fusion of stereo and motion. *PR*, 3(1):121–130, January 1996.
- [16] B.K.P. Horn and B.G. Schunck. Determining optic flow. *Artificial Intelligence*, 17(1-3):185–203, August 1981.
- [17] M. Isaard and J. MacCormick. Dense motion and disparity estimation via loop belief propagation. In *Asian Conference on Computer Vision*, pages 32–41, 2006.
- [18] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, January 2004.

- [19] S. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.
- [20] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the Seventh International Conference on Computer Vision*, pages 674–679, 1981.
- [21] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162, 1992.
- [22] D. Scharstein and R. Szeliski. Middlebury college - stereo vision page research. <http://cat.middlebury.edu/stereo/>.
- [23] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1/2/3):7–42, April-June 2002.
- [24] N. Papenberger T. Brox, A. Bruhn and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, volume 3024, pages 25–36, 2004.
- [25] R.L. Rivest T.H. Cormen, C.E. Leiserson and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 2nd edition, 2001.
- [26] E. Trucco and A. Verri. *Introductory Techniques for 3D Computer Vision*. Prentice Hall Inc., Upper Saddle River, New Jersey, 1998.
- [27] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):475–480, 2005.
- [28] O. Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, August 1999.

- [29] W.D. Wang and J.H. Duncan. Recovering the 3-dimensional motion and structures of multiple moving objects from binocular image flows. *Computer Vision & Image Understanding*, 63:430–446, May 1996.
- [30] A. Waxman and J. Duncan. Binocular image flows: Steps towards stereo-motion fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:715–729, 1986.
- [31] C.Q. Shu Y.Q. Shi and J.N. Pan. Unified optical flow field approach to motion analysis from a sequence of stereo images. *Pattern Recognition*, 27:1577–1590, 1994.