Optimization Algorithms for Discrete Markov Random Fields, with Applications to Computer Vision

Nikos Komodakis

A thesis submitted for the degree of
Doctor of Philosophy

University of Crete
Computer Science Department
Heraklion

Supervisor: Professor Georgios Tziritas

(May 2006)

Abstract

We are what we repeatedly do.
Excellence, then, is not an act, but a habit.

—Aristotle 384 BC-322 BC

A large variety of important tasks in low-level vision, image analysis and pattern recognition can be formulated as discrete labeling problems where one seeks to optimize some measure of the quality of the labeling. For example such is the case in optical flow estimation, stereo matching, image restoration to mention only a few of them. Discrete Markov Random Fields are ideal candidates for modeling these labeling problems and, for this reason, they are ubiquitous in computer vision. Therefore, an issue of paramount importance, that has attracted a significant amount of computer vision research over the past years, is how to optimize discrete Markov Random Fields efficiently and accurately. The main theme of this thesis is concerned exactly with this issue. Two novel MRF optimization schemes are thus presented, both of which manage to extend current state-of-the-art techniques in significant ways.

On one hand, a novel framework is proposed that is based on the duality theory of Linear Programming (LP) and provides an alternative as well as more general view of existing graph-cut methods such as the α-expansion technique, which is included merely as a special case. Moreover, unlike α-expansion which is valid only for MRFs with metric potentials, the derived algorithms provably generate almost optimal solutions for a much wider class of MRFs that are frequently encountered in computer vision, which is an important advance. Results on a variety of low level vision tasks demonstrate the efficacy of our approach.

On the other hand, a novel optimization scheme, called *Priority-BP*, is proposed which carries two very important extensions over standard Belief Propagation

(BP): *priority-based message scheduling* and *dynamic label pruning*. For the first time, these two extensions work in cooperation in order to deal with one of the major limitations of BP: its inefficiency in handling MRFs with very large discrete state-spaces. Moreover, both extensions are generic and do not make any use of domain-specific knowledge. They are therefore applicable to any discrete Markov Random Field i.e. a very wide class of problems in computer vision.

In order to demonstrate the effectiveness of Priority-BP, a novel exemplar-based framework (based on Priority-BP) is also proposed which treats the problems of image completion, texture synthesis and image inpainting in a unified manner, while managing to compare favorably with related state-of-the-art methods. According to our framework, all of the above mentioned tasks are posed as discrete MRF optimization problems with a well-defined global objective function. Thanks to our Priority-BP algorithm, the intolerable (due to the huge number of labels) computational cost of optimizing the resulting MRF is drastically reduced. Furthermore, visually inconsistent results due to greedy patch assignments are avoided, since our method always manages to maintain many candidate source patches for each block of missing pixels. Numerous results on a wide variety of difficult image completion cases prove the efficacy of our framework.

Finally, as an application of our LP-based MRF optimization techniques, we turn our attention to a research topic that lies at the convergence of the fields of computer vision and computer graphics: the virtual reconstruction of 3D environments based on image sequences. Contrary to most of the existing image-based-modeling-and-rendering (IBMR) methods, which typically require large amount of image data and are thus suitable mostly for small scale scenes, here we propose a novel hybrid (geometry & image based) framework that is capable of providing photorealistic walkthroughs of very large, complex outdoor scenes at interactive frame rates. Furthermore, our framework is fully automatic and takes as input only a sparse set of stereoscopic image pairs from the scene. Based on these image pairs, a novel hybrid data representation of a 3D scene, called *morphable 3D-mosaics*, is then automatically extracted. According to it, a 3D scene is represented as a series of enhance local 3D models that allow a continuous morphing between each successive two of them to be taking place during rendering. The morphing is both photometric as well geometric and always proceeds in a phys-

ically valid way. MRFs play a crucial role for the correct estimation of the morphing and, for optimizing these MRFs, we make use of our LP-based optimization methods. Our framework has already been successfully applied to the virtual 3D reconstruction of the Samaria gorge in Crete and a sample from the results that have been obtained is shown as well.

Acknowledgements

I first wish to express my sincere gratitude towards my advisor, Professor Georgios Tziritas, who guided and supported me throughout the years. His office door was always open to me whenever i needed his advice. I also want to thank him for giving me the opportunity to explore and deal with challenging research topics.

I would also like to thank assistant Professor George Georgakopoulos. His wonderful lectures on algorithms and complexity (that i was lucky enough to attend) gave the inspiration for part of the research that was conducted in this thesis. In addition, i would like to thank Professor Panos Trahanias, as well as all the members of my thesis committee, for their insightfull comments, which were indeed of great value for the improvement of the quality and presentation of this thesis.

A special thanks also goes to Professor Nikos Paragios. Although being far away, he has always been willing to offer his valuable advice, as well as its constructive comments, on various early manuscripts.

I should note that the biggest part of the research in this thesis has been funded by the European DHX (Digital Ecological and Artistic Heritage Exchange) project.

Finally, I want to express my deepest gratitude towards my parents for their unfaltering love and support. They have always been a tremendous source of strength for me. Last but not least, i want to thank my wife, Vaggelio. Through all these years she has always been by my side, encouraging, helping and caring. I am pretty sure that, without her support, this thesis would not have been possible.

Nikos Komodakis

Heraklion, 2006.

Contents

Abstract **i**

Acknowledgements **v**

Thesis organization **1**

1 Introduction **5**

1.1 The optimization paradigm in vision 5

1.2 Discrete labeling problems and Markov Random Fields 10

1.3 State-of-the-art optimization methods for discrete Markov Random Fields 17

1.4 Thesis contributions 19

2 Background on Markov Random Fields **25**

2.1 Introduction 25

2.2 Basics of Markov Random Fields 26

2.3 Gibbs random fields and Markov-Gibbs equivalence 28

2.4 *Maximum a posteriori* estimation 30

2.5 State-of-the-art MRF optimization techniques 32

2.5.1 The α-expansion algorithm 32

2.5.2 Loopy belief propagation 35

2.6 Other MRF optimization methods in vision 39

3 Approximate Labeling via Graph Cuts Based on Linear Programming **43**

3.1 Introduction 44

3.2 Related work 50

3.3 The primal-dual schema 52

3.3.1 The primal and dual LPs (Linear Programs) corresponding to Metric Labeling 54

3.3.2 An intuitive view of the dual variables 56
3.3.3 Applying the primal-dual schema to Metric Labeling 57
3.4 The PD1 algorithm 58
3.4.1 Update of the primal and dual variables 63
3.5 The PD2 algorithm 66
3.6 PD3: extending PD2 to the semimetric case 70
3.7 Experimental results 74
3.7.1 Per-instance suboptimality bounds 74
3.7.2 Stereo matching 78
3.7.3 Image restoration and image completion 80
3.7.4 Optical flow estimation 82
3.7.5 Synthetic problems 84
3.8 Conclusions 85

4 Priority-BP and the Problem of Image Completion 87
4.1 Introduction 88
4.2 Image completion as a discrete global optimization problem 94
4.3 Priority-BP 95
4.3.1 Priority-based message scheduling 96
4.3.2 Assigning priorities to nodes 100
4.3.3 Applying Priority-BP to image completion 101
4.3.4 Label pruning 104
4.4 Extensions & further results 106
4.5 Conclusions 111

5 3D Visual Reconstruction of Large Scale Natural Sites 113
5.1 Introduction 115
5.2 Related work 120
5.3 Overview of the modeling pipeline 121
5.4 Local 3D models construction 123
5.4.1 Disparity estimation 124
5.5 Relative pose estimation between successive local models 125
5.5.1 Wide-baseline feature matching under camera looming . . . 126
5.6 Morphing estimation between successive local models 128

5.6.1 Estimating optical flow between wide-baseline images I_k and I_{k+1} . . . 130
5.6.2 Geometric morphing in region $\bar{\Psi}$ . . . 134
5.7 Rendering pipeline . . . 138
5.7.1 Decimation of local 3D models . . . 140
5.8 3D-mosaics construction . . . 142
5.8.1 Rotation (R_{ij}) estimation between views I_i, I_j . . . 143
5.8.2 Geometric rectification of local models . . . 143
5.8.3 Merging the rectified local models . . . 146
5.9 Further results . . . 146
5.10 Conclusions . . . 150

A Technical proofs for theorems of chapter 3 **153**

A.1 Proof of theorem 3.2 about the optimality properties of the PD1 algorithm . . . 153
A.2 Proof of theorem 3.3 about the optimality properties of the PD2$_\mu$ algorithm . . . 160
A.3 Proving the equivalence between algorithm PD2$_{\mu=1}$ and the α-expansion min-cut algorithm . . . 164

References **167**

Author's publication list **181**

List of Figures

0.1 Chart showing thesis overview 1

1.1 An example of the stereo matching problem 7

1.2 In the optical flow problem, we need to compute the 2D pixel displacements between the current and the next frame 7

1.3 In image restoration, we are given an image corrupted with noise and try to recover a restored image that should be as similar as possible to the true original image. 7

1.4 In image segmentation, we ideally want to assign a label to each pixel so that pixels belonging to the same object have equal labels. 8

1.5 For detecting a face inside a given image, we try to locate certain characteristic parts of the face e.g. nose, mouth and eyes. 8

1.6 Typically, for problems in early vision, the graph $\mathcal{G}$ coincides with the image grid e.g. a graph corresponding to a 4×4 image is shown here. 13

1.7 If we try to estimate disparity based only the data terms of the energy function, then a noisy disparity map will almost surely be produced. Here we show the resulting disparity for the left and right image of figure 1.1. To ensure a better result, we must also take contextual constraints into account. 13

1.8 We can model a face as a graph consisting of 4 objects: nose, mouth and two eyes. Each object is assumed to have a specific appearance, while some of the objects are connected to each other with springs. These springs try to keep the objects at an ideal relative distance. For detecting faces, we then need to locate that graph inside the image, but without stressing the springs too much. 17

2.1 The set of nodes in B separates the sets A and C since any path from A to C necessarily passes through B. Therefore, in any MRF with this graph, random variables x_A, x_C will be conditionally independent given variables x_B. 28

2.2 The cliques of the graph in figure (a) are shown in figure (b). Therefore, any Gibbs distribution $p(x)$ on this graph can be expressed as: $p(x) \propto \exp\big(-V_{qrgt}(x_q, x_r, x_g, x_t) - V_{rhg}(x_r, x_h, x_g) - V_{hp}(x_h, x_p) - V_{hs}(x_h, x_s)\big)$ 30

2.3 A cut consisting of the edges sp, pr, rh is shown in this figure. This is, in fact, the minimum st-cut in the above graph since there is no other st-cut with cost less than 10. 33

2.4 The labeling shown on the right image is an example of a green-expansion of the labeling on shown on the left image. In this case, labels correspond to colors. 35

2.5 If a node p wants to send a message $m_{pq}(x_q)$ to a neighboring node q, then it must make use of the messages $m_{sp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from the rest of its neighbors. 37

2.6 If a node p wants to calculate its belief $b_p(x_p)$ about any of the labels $x_p \in \mathcal{L}$, it must then collect the messages $m_{sp}(x_p), m_{qp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from all of its neighboring nodes. 38

3.1 **(a)** The difficulty of the ML problem depends critically on the type of label distance d_{ab} chosen. The global optimum in the case of a linear distance function can be found by using the technique described in [68], while an approximate solution in the case of a metric distance can be computed using the α-expansion method [27]. **(b)** A comparison of our framework with respect to existing state-of-the-art graph-cut methods. 48

3.2 **(a)** By weak duality the optimal cost $c^T x^*$ will lie between the costs $b^T y$ and $c^T x$ of any primal-dual pair of feasible solutions (x, y). Therefore if $b^T y$ and $c^T x$ are close enough (e.g. their ratio r_1 is $\leq f$) so are $c^T x^*$ and $c^T x$ (e.g. their ratio r_0 is $\leq f$ as well), thus proving that x is an f-approximation to x^*. **(b)** Dual and primal feasible solutions make local improvements to each other until the final costs $b^T y^t$, $c^T x^t$ are close enough (e.g. their ratio is $\leq f$). We can then apply the primal-dual principle and conclude that x^t is an f-approximation to x^*. 53

3.3 Visualization of the dual variables for a graph G consisting of just 2 neighbors p, q while $L = \{a, b, c\}$. Each vertex holds a copy of all labels in L and all these labels are represented by circles which are located at certain heights specified by the ht variables. Label c at p is pulled up due to the increase of the balance variable $y_{pq,c}$ and so the corresponding label at neighboring vertex q is pulled down due to the decrease of the conjugate variable $y_{qp,c}$. The active labels of p, q are drawn with a thicker circle. 57

3.4 The basic structure of the algorithms PD1, PD2 and PD3. 57

3.5 **(a)** An arrangement of labels (represented by circles) for a graph G with 3 vertices p, q, r and 2 edges pq, qr of weights w_{pq}, w_{qr}. The label set is $L = \{a, c\}$. The thicker circles represent the active labels. Also, the red arrows indicate how the c labels will move in respond to an update of the balance variables while the dashed circles show the final position of those labels after the update. **(b)** The corresponding graph $G_c^{x,y}$ that will be used for updating the balance variables. Interior/exterior edges are drawn with solid/dashed lines respectively. 61

3.6 **(a)** An initial arrangement of labels' heights at the start of a c-iteration. All vertices p, q, r are currently assigned label a (as indicated by the thick circles). **(b)** The new heights as updated by the UPDATE_DUALS_PRIMALS routine after applying a maximum flow algorithm to the graph in (d). Red and blue arrows show how the c-labels move due to the update of the balance variables. Movements due to changes in conjugate balance variables are drawn with the same line style and color. The dashed circles indicate the final positions of the c-labels. **(c)** The new active labels (thick circles) that were selected based on the "reassign rule". Only vertex p had to change its active label into c. This is so because only p still has its label c below its previous active label a or equivalently only edge sp of the graph in (d) is unsaturated while any paths to q or r are not. **(d)** The associated capacitated graph (assuming that all balance variables are initially zero) and the resulting flows due to the maximum flow algorithm. Notice that, as expected, the flows f_p, f_q, f_r at exterior edges reflect the total movement of the c-labels at p, q, r respectively. In this example the Potts metric has been used as distance d_{ab} (i.e. $a \neq b \Rightarrow d_{ab} = 1$). 65

3.7 Pseudocode for the PD1 algorithm. 67

3.8 Pseudocode of the PD2$_\mu$ algorithm. The routine UPDATE_DUALS_PRIMALS is not shown because it is the same as the corresponding routine of the PD1 algorithm. The only difference is that a subset of the edges of $G_c^{x^k, \bar{y}^k}$ have different capacities (see text). 68

3.9 **(a)** The left and **(b)** right images for one stereo pair from the Tsukuba data set. **(c)** The disparity estimated by the PD1 algorithm. **(d)** and the PD2$_{\mu=1}$ algorithm. The Potts distance (a metric) has been used in this example and so PD3$_a$, PD3$_b$, PD3$_c$ produce the same result with PD2$_{\mu=1}$. 76

3.10 These 3 plots show how the primal-dual ratios vary during the first 4 outer iterations (or equivalently the first $60 = 4 \cdot 15$ inner iterations) using the Tsukuba sequence as input. **(Left)** The potts metric, **(Middle)** the trunc.linear metric and **(Right)** the trunc. quad. semimetric have been used respectively as label distance d_{ab}. Notice how rapidly the ratios drop in all cases (i.e. they get very close to 1 just after a few inner iterations). 79

3.11 **(a)** One image from the SRI tree image sequence. **(b)** Computed disparities when using PD3$_a$ and the semimetric $d^{\kappa,\lambda}_{4,ab}$ with $(\kappa, \lambda) = (2, 10)$. **(c)** Disparities computed by the α-β-swap algorithm using the same semimetric. The solution of α-β-swap has 8.3% higher energy than the corresponding solution of the $PD3_a$ algorithm despite the fact that both algorithms try to minimize exactly the same energy function. 79

3.12 **(a)** Uniqueness constraint is not favored and the graph G coincides with the image grid. A common label distance is used for all edges (i.e. $hdist^1 = vdist^1$) **(b)** To favor the uniqueness constraint we introduce additional horizontal edges in G which connect any pixel with the K pixels to its right (K is the maximum disparity). 80

3.13 Red pixels indicate occlusions 80

3.14 **(a)** Original uncorrupted image. **(b)** Noisy input image. **(c)** Restored image using semimetric $d^{\kappa,\lambda}_{4,ab}, (\kappa, \lambda) = (2, 30)$ **(d)** Restored image using the truncated linear metric $d^{\lambda}_{2,ab}, \lambda$=30. 81

3.15 Examples of image restoration and image completion 82

3.16 Estimated flow between frames 4, 5 (1^{st} row) and 11, 12 (2^{nd} row) of *yosemite* sequence. Although more outer iterations were used by α-β-swap, its optical flow had 19.2% and 56.7% higher energy than our optical flow. 83

3.17 α-β-swap produces an energy which is higher by **(a)** 17%, **(b)** 23% and **(c)** 28% with respect to our algorithm's energy. Notice that as the number of labels increases the gap in performance increases as well. 85

3.18 A synthetic example where the graph G has 3 vertices $\{p, q, r\}$ and 2 edges $\{pq, qr\}$ while the labels L are $\{a, b, c\}$. Label costs c_{pa} and the distance d_{ab} (a semimetric) are shown. The α-β-swap algorithm can get stuck in labeling A whose cost is T i.e. arbitrarily larger than the true minimum cost which is 4 (labeling B). On the contrary $PD3_a$, $PD3_b$, $PD3_c$ can always locate the optimal labeling B. Example taken from [27]. 85

4.1 *Object removal* is just one example of the many uses of image completion. In the specific example shown above, the user wants to remove a person from the input image on the left. He therefore simply marks a region around that person and that region must then be filled automatically so that a visually plausible outcome is obtained. 89

4.2 Image inpainting methods, when applied to large or textured missing regions, oversmooth the image and introduce blurring artifacts. . 91

4.3 The nodes and edges of an MRF associated with image completion. In this example, the w, h parameters were set equal to $w = 2gap_x$, $h = 2gap_y$. 94

4.4 For the boundary node r, its label cost $V_r(x_r)$ will be an SSD over the red region while for nodes p, q their potential $V_{pq}(x_p, x_q)$ will be an SSD over the green region. Node s is an interior node and so its label cost $V_s(x_s)$ will always be zero. 94

4.5 Message scheduling during the forward pass: currently only red nodes have been committed and only messages on red edges have been transmitted. Among uncommitted nodes (*i.e* blue nodes) the one with the highest priority (*i.e* node p) will be committed next and will also send messages only along the green edges (*i.e* only to its uncommitted neighbors q, r). Messages along dashed edges will be transmitted during the backward pass. Priorities are indicated by the numbers inside uncommitted nodes. 100

4.6 In column (c) darker patches correspond to nodes that are visited earlier during message scheduling at the first forward pass 102

4.7 The plots in (a), (b) and (c) show the sorted relative beliefs for the MRF nodes a, b and c in figure (d) at the start of Priority-BP. Relative beliefs plotted in red correspond to labels in the confusion set. This set determines the priority of the corresponding node. 103

4.8 **(a)** Although the red, green and blue patches correspond to distinct labels, they are very similar and so only one has to be an active label for a node. **(b)** A map with the number of active labels per node (for the 2^{nd} example of Figure 4.6). Darker patches correspond to nodes with fewer labels. As can be seen interior nodes often require more labels. **(c)** The corresponding histogram showing the percentage of nodes using a certain number (in the range $L_{min} = 3$ to $L_{max} = 20$) of active labels. **(d)** The active labels for node a in Fig. (a). 105

4.9 Image completion. From left to right: original images, masked images, visiting order at 1^{st} forward pass, Priority-BP results 107

4.10 texture synthesis results produced with the Priority-BP algorithm 108

4.11 An example of text removal 109

4.12 An image inpainting example 109

4.13 Texture synthesis using the "incoherence penalty terms". Notice that, in this case, the output texture has been synthesized by copying large chunks from the input texture. 109

4.14 Some more results on image completion, produced using the Priority-BP algorithm. From left to right: original images, masked images, visiting order at 1^{st} forward pass, Priority-BP results 112

5.1 A schematic view of the plenoptic function 116

5.2 Overview of our approach: **(a)** A sparse set of stereoscopic views is captured at key-positions along the path **(b)** One local 3D model is constructed out of each stereoscopic view **(c)** As the user traverses the path a morphable 3D model is displayed during rendering. This way a continuous morphing between successive local models takes place at any time, with this morphing being both photometric as well as geometric. 117

5.3 The modeling pipeline 121

5.4 For calibrating our camera we capture images of a chess pattern at random positions and orientations. 122

5.5 **(a)** Depth map Z_0 of a local model (black pixels do not belong to its valid region dom_0). **(b)** A rendered view of the local model using an underlying triangle mesh 124

5.6 The 2 stages needed for disparity estimation 125

5.7 **(a)** Image I_k along with computed optical flow vectors (blue segments) for all points marked white. **(b)** Image I_{k+1} along with matching points (also marked white) for all marked points of (a). A few epipolar lines are also shown. In both images, the yellow square around a point is analogous to the point's estimated scale factor (10 scales $S = \{1, 0.9^{-1}, ..., 0.1^{-1}\}$ have been used). 127

5.8 Two more examples (one example per row) of wide baseline matching from another scene. Optical flow vectors (on image I_k) as well as estimated epipolar lines (on image I_{k+1}) are shown again. Also, notice the large camera motion taking place in the top example e.g. the stones in the water appear much closer to the camera in figure (b) than in figure (a). 127

5.9 Given a point $p \in I_k$ and a candidate matching point $q \in I_{k+1}$, we search across a range of scales $1{=}s_0{<}s_1{<}...{<}s_n$ by first projecting q on rescaled images and then comparing the neighborhood of each of the resulting pixels with the neighborhood of p in I_k. 133

5.10 Maps of: **(a)** *scale factors* and **(b)** *optical flow magnitudes* for all points in Ψ, as estimated after applying the optical flow algorithm to the images of Fig. 5.7 and while using 10 possible scales $S = \{1, 0.9^{-1}, ..., 0.1^{-1}\}$. **(c)** Corresponding optical flow magnitudes when only one scale $S = \{1\}$ has been used. As expected, in this case the algorithm fails to produce exact optical flow for points that actually have larger scale factors. We note that darker pixels in a grayscale image correspond to smaller values. 133

5.11 **(a)** Destination depth map Z_{dst} for points inside region Ψ after using optical flow of Fig. 5.10(b) and applying eq. (5.3). To completely specify morphing we need to extend this map to the points in region $\bar{\Psi}$ **(b)** Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ without applying geometric morphing. Notice that there exist discontinuities along the boundary $\partial\bar{\Psi}$. **(c)** Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ after applying geometric morphing. 134

5.12 Rendered views of the morphable 3D-model during transition from the key-position corresponding to image 5.7(a) to the key-position of image 5.7(b): **(a)** when no geometric morphing is applied to points in $\bar{\Psi}$ and **(b)** when geometric morphing is applied to points in $\bar{\Psi}$. **(c)** A close-up view of the rendered image in (b). Although there is no geometric discontinuity, there is a difference in texture resolution between the left part of the image (points in $\bar{\Psi}$) and the right part (points in Ψ) because only points of the latter part are morphed photometrically. 135

5.13 Pixel shader code (and the associated vertex shader code), written in GLSL (OpenGL Shading Language), for implementing the photometric morphing. 138

5.14 Skeleton code in C for applying vertex blending in OpenGL. . . . 138

5.15 **(a)** Estimated disparity field corresponding to a local 3D model L_k. **(b)** Resulting full 3D model produced when a non-decimated 2D triangulation of the geometric maps has been used. **(c)** Simplified 3D model of L_k produced using a decimated 2D triangulation where the e_{max} threshold has been set equal to 0.5 pixels. 141

5.16 Rendered views of: **(a)** a local model L_j **(b)** a local model L_i **(c)** a 3D-mosaic of L_i, L_j without geometry rectification (holes are due to errors in the geometry of the local models and not due to misregistration in 3D space) **(d)** a 3D-mosaic of L_i, L_j after $\mathrm{RECTIFY}_{L_i}(L_j)$ has been applied **(e)** a bigger 3D-mosaic created from L_i, L_j as well as another local model which is not shown 145

5.17 A synthetic example illustrating the superiority of our approach against feathering (see also text). **(a)** True depth maps. **(b)** Estimated noisy depth maps. **(c)** Resulting 3D-mosaic's depth map using feathering. **(d)** Resulting 3D-mosaic's depth map using our method. 146

5.18 Another example of a 3D-mosaic constructed using our method. **Top row:** three separate local 3D-models **Bottom row:** the resulting 3D-mosaic 147

5.19 Two stereoscopic views as would be rendered by the VR system (for illustration purposes these are shown in the form of red-blue images). 148

5.20 Each row contains sample rendered views of a separate morphable 3D model. In each row the leftmost, rightmost images correspond to the views at pos_k, pos_{k+1} respectively. 149

5.21 Some rendered views that are produced as the virtual camera traverses a path through the so-called "Iron Gates" area, which is the most famous part of the Samaria Gorge. In this case the virtual camera passes through a series of successive morphable 3D models. 149

5.22 **(a)** The left image of a stereoscopic image pair that has been captured at a region passing through a small river. **(b)** The estimated disparity by using a stereo matching procedure. As expected, the disparity field contains a lot of errors for many of the points on the water surface. This is true especially for those points that lie near the sun reflections on the water. **(c)** The corresponding disparity when a 2D homography is being used to fill the left-right correspondences for the points on the water. In this case the water surface is implicitly approximated by a 3D plane. 151

5.23 **(a)** Some rendered views of the gorge that also contain a synthetically generated volumetric fog. **(b)** A rendered view where a synthetic 3D model of the *agrimi*, a wild animal which is specific to the Samaria Gorge, has been integrated into the 3D virtual environment. **(c)** Another view with an oleander plant integrated as well. 151

List of Tables

3.1 Average suboptimality bounds (colums 2-6) obtained for the Tsukuba data set. As expected they are much closer to 1 than the theoretical suboptimality bounds f_{app} listed in the last column and thus a nearly optimal solution is obtained in all cases. Note that $PD2_{\mu=1}$ can be applied only if d_{ab} is a metric and in that case PD2$_{\mu=1}$, PD3$_a$, PD3$_b$ and PD3$_c$ (as well as their bounds) coincide. 77

3.2 The average suboptimality bounds (columns 2-4-6-8-10) obtained when applying our stereo matching algorithms to one scanline at a time (instead of the whole image). In this case, we are also able to compute the true average suboptimality (columns 3-5-7-9-11) of the generated solutions using dynamic programming. As can be seen by inspecting the table the suboptimality bounds always approximate the true suboptimality relatively well, meaning that they can be safely used as a measure for judging the goodness of a generated solution. 77

3.3 Error statistics for the image restoration example of Fig. 3.14 . . . 81

Thesis organization

An overview of this thesis appears in Figure 0.1. Its main theme is concerned with robust as well as efficient optimization techniques of discrete Markov Random Fields. To this end, two novel optimization schemes are proposed, both of which manage to extend current state-of-the-art techniques in significant ways. As a demonstration of the effectiveness of our algorithms, we also use them in two different computer vision applications: image completion and visual 3D reconstruction of large scale natural sites. In fact, for both of these tasks, novel frameworks are proposed, which also nicely advance related state-of-the-art techniques.

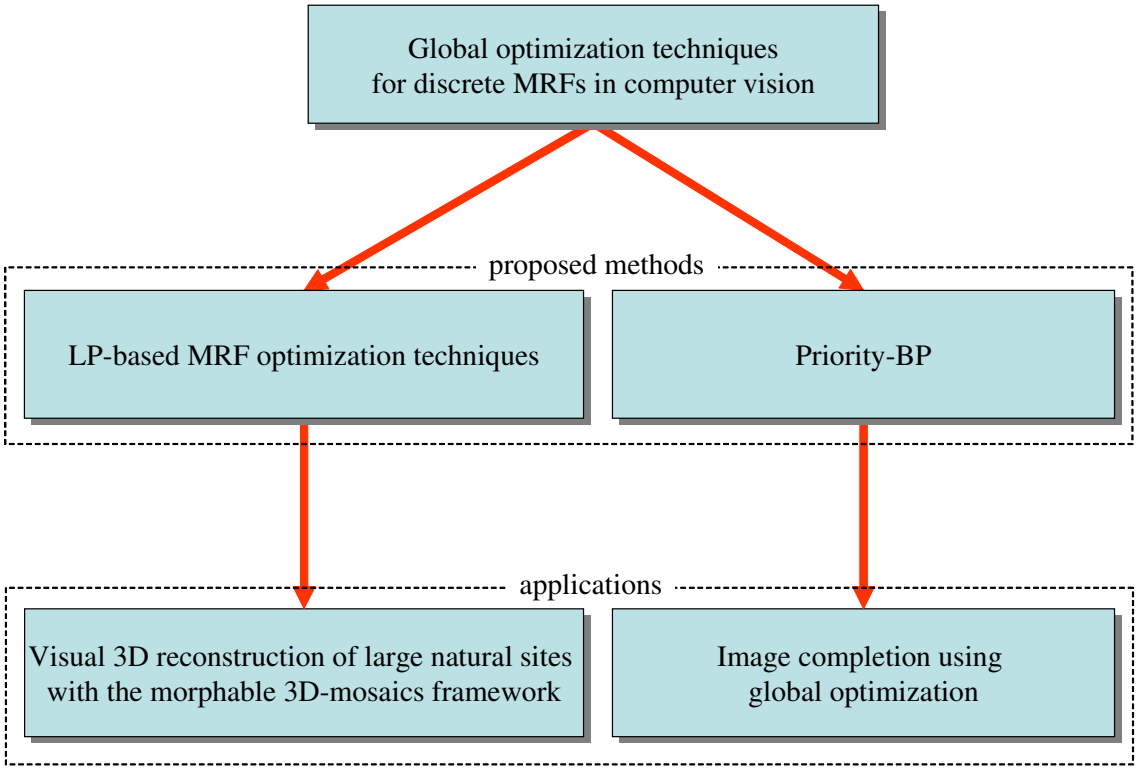


Fig. 0.1: Chart showing thesis overview

Based on the observations made above, this thesis is organized (in a chapter by chapter basis) in the following manner:

Chapter 1

This chapter introduces the reader to the optimization paradigm in vision and sets the general context for the optimization framework that will be used in the remainder of the sequel. By providing examples, it illustrates the modeling power of MRFs and explains how these can capture a wide range of problems in computer vision. Finally, it provides a brief description of the main contributions of this thesis with respect to the current state-of-the-art MRF optimization methods.

Chapter 2

This chapter provides all necessary background about Markov Random Fields and gives a bayesian justification for the form of their energy function. In addition, the two current state of the art MRF optimization techniques, α-expansion and loopy belief propagation, are briefly described, while a short review of some other popular MRF optimization techniques is provided as well.

Chapter 3

During this chapter, a novel framework is presented that is based on the duality theory of linear programming and generalizes as well as extends existing graph-cut methods. A number of algorithms are derived based on our framework, just one of which is shown to be equivalent to the α-expansion technique. Moreover, unlike α-expansion, it is proved that the proposed algorithms can generate almost optimal solutions for a very wide class of Markov Random Fields with both metric and non-metric potential functions.For demonstrating the effectiveness of our framework, we also present experimental results on a variety of low level vision tasks.

Chapter 4

During this chapter, a novel MRF optimization scheme, called Priority-BP, is presented which carries 2 very important extensions over standard belief propagation: priority-based message scheduling and dynamic label pruning. Together,

these two extensions manage to resolve what is currently considered as the major limitation of Belief Propagation: its inefficiency in handling MRFs with very large discrete state-spaces. Our method is generic and can therefore be applied to any MRF i.e. a very wide class of problems in computer vision. To demonstrate the effectiveness of Priority-BP, we show how that algorithm can be applied to the problems of image completion, texture synthesis and image inpainting. A novel exemplar-based framework is thus presented that manages to unify all all of the above mentioned tasks by formulating them as discrete MRF optimization problems with a well defined objective function. Results on a wide variety of difficult image completion examples are included and demonstrate the efficacy of our approach.

Chapter 5

During this chapter, as an application of our LP-based MRF optimization techniques, we turn our attention to the study of novel image based modeling and rendering methods, a research topic that lies at the convergence of computer vision and graphics. To overcome the limitations of existing IBR methods, that usually require large amount of image data and are thus targeted mostly for small scale scenes, a novel hybrid (geometry & image based) framework is presented which is capable of providing photorealistic walkthroughs of large, complex outdoor scenes at interactive frame rates. The proposed framework is called *morphable 3D-mosaics* and its main ingredients are thoroughly explained and analyzed during this chapter. In addition, we show results that have been obtained by successfully applying our IBMR framework to the virtual 3D reconstruction of the Samaria gorge in Crete, which is one of the largest and most magnificent gorges in Europe.

Appendix A

This appendix contains all technical proofs for the theorems and lemmas presented during chapter 3.

CHAPTER 1

Introduction

This chapter servers an introductory goal. On one hand, it introduces the optimization paradigm and explains why optimization techniques are so important in computer vision research. On the other hand, it illustrates through examples how such techniques can be used for solving a wide variety of computer vision problems. Furthermore, it sets the general context for the optimization framework that will be used throughout the rest of the thesis, while it also briefly motivates the reason why new optimization methods, capable of handling the challenges posed by current computer vision problems, are needed. Finally, the chapter ends by providing a brief description of the main contributions of this thesis with respect to state-of-the-art optimization algorithms that are currently used in computer vision research.

Vision is the art of seeing things invisible

—*Jonathan Swift*

1.1 The optimization paradigm in vision

As electronic components get cheaper and cheaper, there is a dramatic increase in the use of computer technology over the last years. Nowadays computers assist humans for a large variety of tasks. They are used e.g. in communications (computer networks), in automating laborious tasks (car industry) or even just for entertainment. However, despite this fact, the ability for computers to mimic the human intelligence remains still an elusive goal. A first step towards achieving this goal would be made if computers could take advantage of (i.e. analyze) the huge amount of visual data that can be given as input to them (e.g. by connecting them to peripheral devices such as digital cameras) and which contain numerous

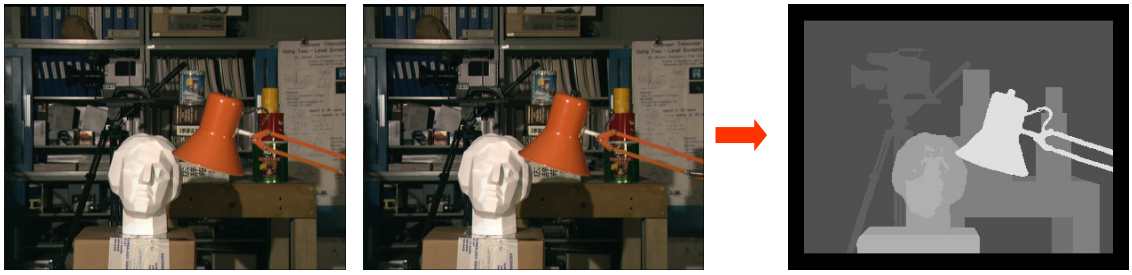
and valuable information about the surrounding environment. This is exactly the main objective of the field called Computer Vision: to make computers "see", i.e. to make computers capable of interpreting and analyzing digital images (either static or moving).

To this end, computer vision needs to deal with a variety of interrelated tasks, each one of them being responsible for extracting a different type of semantic information that is hidden behind the raw image data. To give to the reader a rough idea of what kind of tasks one may encounter in computer vision, we mention here only a few typical examples of computer vision problems that are related to extracting either low-level or high-level information:

Stereo matching [18, 90, 115, 126, 139, 151] We are given a pair of two images, a left and a right image, captured by two digital cameras, which are aligned to each other, i.e. they are located at the same height and both look towards the same direction. We want to find for each pixel of the left image its horizontal displacement, also known as *disparity*, in the right image (see Figure 1.1). This way, 3D information about the surrounding scene can be extracted (e.g. by intersecting the 3D rays that pass through corresponding pixels in the two images). This mimics the ability of the human vision system to extract 3D information based on the stereoscopic images entering the left and right eye.

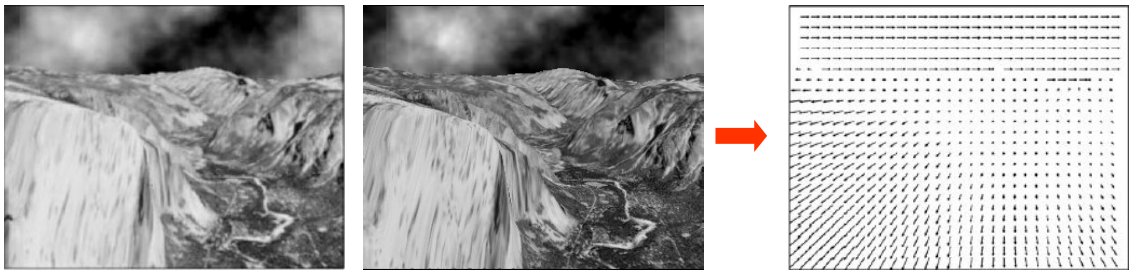
Optical flow estimation [9, 62, 65, 67, 141] Given two successive frames from an image sequence that has been captured by a moving video camera, we want to find for each pixel in the current frame where it has moved in the next frame. In this case, a pixel may appear to move not only horizontally but also vertically and so we need to estimate the 2D displacement, called *optical flow*, for each pixel (see Figure 1.2).

Image restoration or denoising [33, 88, 93, 104, 108, 122] We are given a digital image corrupted with noise (e.g. due to bad lighting conditions at the time of the capture). Before trying to infer any higher level information from that image, it would be necessary to remove the noise and restore the original content of the image (see Figure 1.3). In other words, we seek to find the true underlying pixel intensities (or colors if we are dealing with color images).



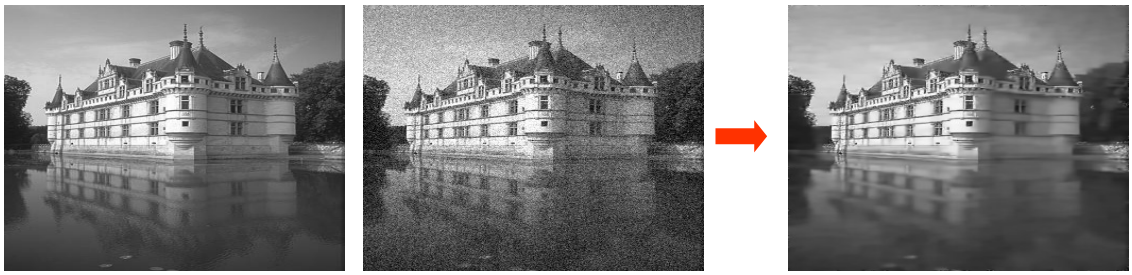
(a) Left image (b) Right image (c) True disparity map

Fig. 1.1: An example of the stereo matching problem



(a) Current frame (b) Next frame (c) True optical flow field

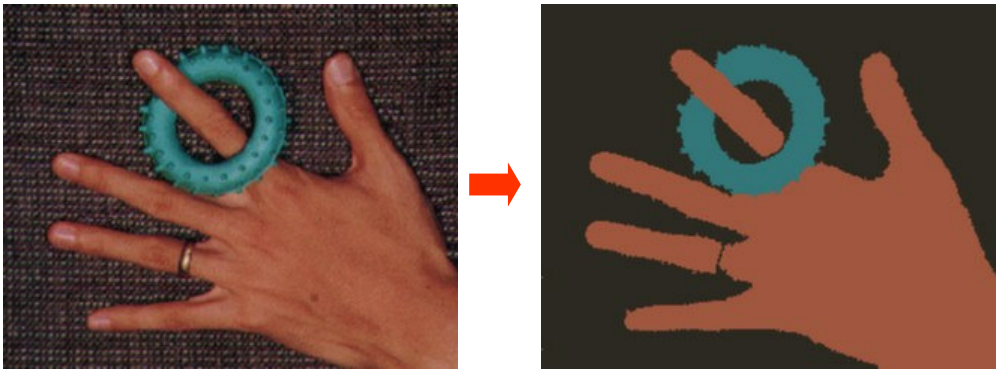
Fig. 1.2: In the optical flow problem, we need to compute the 2D pixel displacements between the current and the next frame



(a) True original image (b) Image corrupted with noise (c) Restored image

Fig. 1.3: In image restoration, we are given an image corrupted with noise and try to recover a restored image that should be as similar as possible to the true original image.

Image segmentation [48, 94, 119, 144, 148] Given as input a single image we want to be able to partition it into regions so that each region corresponds to one of the different object that may be present in that image. Put otherwise, we want to assign a label to each pixel so that pixels corresponding to the same object have the same label (see Figure 1.4). The image segmentation task can be useful if we wish to extract further higher-level information for each one of the objects displayed in the image.



(a) Input image (b) Segmented image: different colors correspond to different labels

Fig. 1.4: In image segmentation, we ideally want to assign a label to each pixel so that pixels belonging to the same object have equal labels.

(a) Input image (b) Detected parts of a face

Fig. 1.5: For detecting a face inside a given image, we try to locate certain characteristic parts of the face e.g. nose, mouth and eyes.

Object detection and localization [2, 30, 46, 47, 142] In this case, we want to know if an object, which belongs to a particular category of objects (e.g. people faces), is located inside a given image and, in case it is, we also want to find out its 2D location. To this end, one possible strategy is to use a part based representation of the object and then try to locate its various parts inside the image. For instance, for detecting faces one might start looking for four parts: one nose, one mouth and two eyes (see Figure 1.5). Of course, in this case, care must be taken so that the spatial configuration of the detected parts does resemble the typical spatial configuration of a person's nose, mouth and eyes.

In all of the tasks mentioned above we want to infer some hidden quantities $\vec{x}$ (disparity, optical flow, etc.) based on some (possibly) noisy observations $\vec{d}$ consisting of the input visual data. One common way to achieve this is by reformulating the task at hand as an optimization problem. In particular, we design an objective function $\mathcal{F}$ (also called the energy function):

$$\mathcal{F} : (\vec{x}, \vec{d}) \rightarrow \mathcal{F}(\vec{x}, \vec{d})$$

which assigns an energy $\mathcal{F}(\vec{x}, \vec{d})$ to each possible combination $(\vec{x}, \vec{d})$ of the hidden quantities and the input data. This energy should represent how bad any given solution $\vec{x}$ fits the input visual data $\vec{d}$ as well as how bad $\vec{x}$ satisfies all physical constraints imposed by the problem at hand. Based on this framework, we then choose that solution $\vec{x}_{opt}$, which yields the minimum energy among all possible $\vec{x}$:

$$\vec{x}_{opt} = \arg\min_{\vec{x}} \mathcal{F}(\vec{x}, \vec{d})$$

The extensive use of the optimization paradigm in vision is favored by the fact that various uncertainties exist in almost every vision process. For instance, the sources of uncertainties can be image noise (due to imperfect sensors or quantization errors), occlusions in the observed image or ambiguities in the visual interpretation. Based on this fact, one can quickly realize that *perfect or exact solutions hardly exist*. On the contrary, inexact but optimal solutions are usually what one can hope to obtain in the best case. In fact, it is exactly due to the existence of these uncertainties that principles from statistics or probability theory are often used as the basis for deriving the exact form of the energy function $\mathcal{F}(\cdot, \cdot)$. Furthermore, one additional advantage of the optimization approach to vision is coming from the fact that the energy function itself can always provide a global quantitative measure that can be used for evaluating the goodness of a resulting solution. Moreover, this function $\mathcal{F}(\cdot, \cdot)$ can be also used as a guide during the search for a minimal solution.

We should note that for any given instance of a computer vision problem, the observations $\vec{d}$ remain fixed (i.e. the optimization is taking place only with respect to the hidden quantities $\vec{x}$). Therefore, for notational simplicity, we will hereafter drop the symbol $\vec{d}$ from $\mathcal{F}(\vec{x}, \vec{d})$ and denote the objective function simply as $\mathcal{F}(\vec{x})$.

Furthermore, $\vec{x}$ will always be assumed to be a vector and we will therefore refer to it simply as x hereafter.

1.2 Discrete labeling problems and Markov Random Fields

Two of the most important issues in optimization-based vision are:

- on one hand, the **modeling of the state of nature**, i.e. how do we choose to represent the hidden quantities x
- and on the other hand, the **objective function**, i.e. what kind of form do we allow the energy function $\mathcal{F}(\cdot)$ to have

These two issues are interrelated to each other and can have a great impact on how effective the optimization process will be.

With respect to the first of these issues, a very common way of representing the hidden quantities is through a discrete set of labels, in which case the problem is reduced to a discrete labeling problem. More specifically, whenever we refer to this term we will hereafter mean a problem which is defined in terms of two basic entities: a graph $\mathcal{G}$ and a set of labels $\mathcal{L}$.

The graph $\mathcal{G} = (V, E)$ consists of a discrete set of N objects[1] and a set of edges E. The objects will hereafter be denoted as:

$$V = \{p, q, r, \ldots\}$$

and can represent features, e.g. a corner point or a line segment, on which a quantity must be estimated, while the edges E of the graph are used for encoding all relationships between the objects of $\mathcal{G}$. Typically, for problems in low level vision, objects will correspond to pixels in the image grid.

Furthermore, in our case, it will always be assumed that the set of labels $\mathcal{L}$ is discrete, consisting of $|\mathcal{L}|$ labels in total. These labels will correspond to the hidden quantities that we want to estimate and can represent e.g. intensities, disparities, or any other quantity of interest. Under these circumstances the

[1] we will use the terms object, vertex, node, site interchangeably throughout

problem is reduced to that of assigning a label from the label set $\mathcal{L}$ to each one of the objects in V. In other words we need to define a mapping x with domain $\mathcal{L}$ and range V, i.e.:

$$x : \mathcal{L} \rightarrow V$$

so that $x_p = x(p)$ represents the unique label assigned to node x_p.

With respect to the issue of specifying the form of the energy function $\mathcal{F}(\cdot)$ a very important aspect is related to the fact that the chosen energy function should be able to somehow encode all *contextual constraints* between the objects. Over the years, it has been realized by computer vision researchers that this is an absolutely necessary condition for one to be able to interpret the visual information hidden in images [85, 105, 132]. It turns out that a very good (and common) way for modeling these contextual constraints is by using what is currently known as a discrete *Markov Random Field* (MRF) [87, 147]. Markov Random Fields have their roots in probability theory and are thus a particular type of probabilistic graphical model. More specifically, they belong to the class of the so-called undirected graphical models. A more rigorous description of Markov Random Fields will be given in chapter 2 along with a justification of the form of their associated objective function. For now, however, it suffices to state directly what the exact form of that objective function $\mathcal{F}(\cdot)$ will be [2]:

$$\mathcal{F}(x) = \mathcal{F}_{data}(x) + \mathcal{F}_{prior}(x).$$

The first term is known as the *data term* and is decomposed in the following way:

$$\mathcal{F}_{data}(x) = \sum_{p \in V} V_p(x_p),$$

while the second term, which is also known as the *prior term*, is defined to be equal to:

$$\mathcal{F}_{prior}(x) = \sum_{(p,q) \in E} V_{pq}(x_p, x_q).$$

[2] Throughout this thesis we are assuming a 1^{st} order discrete Markov Random Field containing only pairwise interactions between objects. Note however that this does not hurt the generality of our formulation, since even a MRF of higher order (i.e. a MRF involving higher order interaction terms) can be reduced to a 1^{st} order MRF by introducing a set of auxiliary variables [145].

The general form of the objective function is thus the following:

$$\mathcal{F}(x) = \sum_{p \in V} V_p(x_p) + \sum_{(p,q) \in E} .V_{pq}(x_p, x_q) \tag{1.1}$$

Intuitively, the data term $V_p(x_p)$ encodes how much the assignment of label x_p to node p disagrees with the observed image data at that node. On the other hand, the prior terms $V_{pq}(x_p, x_q)$ are used for encoding the contextual constraints that need to be imposed between neighboring objects in the graph. Put otherwise, they express our a priori knowledge about the labels before even observing any image data at all. They are therefore used for assigning a high energy to any labeling which is considered to be highly unlikely a priori. The functions $V_p(\cdot)$, $V_{pq}(\cdot, \cdot)$ are also known as single and pairwise *potential functions* respectively.

Despite this seemingly simple formulation of the energy function associated to an MRF, Markov Random Fields are capable of capturing a very wide range of problems in computer vision. This will become obvious in the illustrating examples that now follow:

Stereo matching:

In this case the labels correspond to a discretized set of disparities. We therefore set:

$$\mathcal{L} = \{0, 1, ..., K\},$$

where K denotes the maximum allowed disparity.

The graph $\mathcal{G} = (V, E)$ corresponds to the image grid, i.e. each object represents a pixel and has the top, bottom, left and right pixel as its neighbors (see Figure 1.6).

The data terms should measure how well corresponding pixels in the left and right image (as determined by the disparity field) match, i.e. how similar their intensities are. The simplest such measure is:

$$V_p(x_p) = |I_{right}(p - x_p) - I_{left}(p)|$$

where the symbols I_{left} and I_{right} represent the 2D array of intensities for the left and right image respectively.

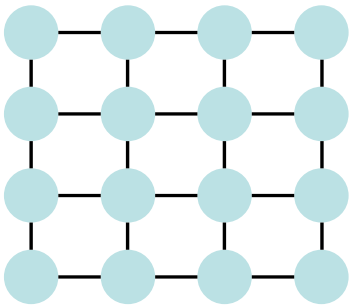


Fig. 1.6: Typically, for problems in early vision, the graph $\mathcal{G}$ coincides with the image grid e.g. a graph corresponding to a 4×4 image is shown here.

Fig. 1.7: If we try to estimate disparity based only the data terms of the energy function, then a noisy disparity map will almost surely be produced. Here we show the resulting disparity for the left and right image of figure 1.1. To ensure a better result, we must also take contextual constraints into account.

If the disparity estimation is based only on the optimization of the data terms, then a rather noisy result will be produced (see Figure 1.7). To fix this, we need to impose a so-called *smoothness* contextual constraint. This generic constraint, which is typically used in many problems in computer vision, merely expresses the fact that physical properties in a neighborhood of space or time present some coherence and do not change in an abrupt fashion. In our case, this translates to the a priori knowledge that pixels which lie next to each other in the image plane tend to have similar disparities most of the time (except for pixels lying along an object's boundary). To this end, the following pairwise potential function will be used, which penalizes neighboring pixels with unequal disparities and is also known as

the Potts model in the literature [26]:

$$V_{pq}(x_p, x_q) = \begin{cases} 1, & \text{if } x_p \neq x_q \\ 0, & \text{if } x_p = x_q \end{cases}$$

Optical flow estimation:

The difference with stereo matching is that now labels are two dimensional vectors, representing displacements in both the horizontal and vertical image axis:

$$\mathcal{L} = \{-K_x, ..., K_x\} \times \{-K_y, ..., K_y\},$$

where K_x and K_y represent the maximum allowed horizontal and vertical displacements in pixels.

The graph $\mathcal{G}$ as well as the data terms $V_p(\cdot)$ are defined in the same way as in stereo matching. E.g.:

$$V_p(x_p) = |I_{next}(p + x_p) - I_{current}(p)|,$$

where $I_{current}$ and I_{next} represent the 2D array of intensities for the current and next frame respectively.

The prior terms should again impose a smoothness constraint for the 2D optical flow field and can be defined as follows:

$$V_{pq}(x_p, x_q) = \min(||x_p - x_q||^2, M)$$

Here, M denotes the maximum allowed penalty that can be imposed. The main use of this parameter is for not allowing discontinuities along object boundaries to be overpenalized [19].

Image restoration:

The labels correspond to image intensities:

$$\mathcal{L} = \{0, 1, ..., 255\}.$$

The graph $\mathcal{G}$ again coincides with the image grid.

The data terms are defined so as to express the fact that the restored intensity x_p at any pixel p should be close to the observed intensity $I(p)$, i.e.:

$$V_p(x_p) = |I(p) - x_p|$$

where I represents the 2D array of observed image intensities.

A pairwise potential function which is commonly used in image restoration is the truncated quadratic semimetric:

$$V_{pq}(x_p, x_q) = \min(|x_p - x_q|^2, M)$$

where M denotes again the maximum penalty that can be imposed.

Image segmentation:

Here, each segmented region corresponds to a different label. E.g., in the case of binary segmentation where we want to partition the image into a foreground and a background region, we set:

$$\mathcal{L} = \{\text{BACKGROUND}, \text{FOREGROUND}\}$$

Graph G coincides with the image grid.

The data terms measure the log likelihood of a pixel belonging to either the foreground or the background:

$$V_p(\text{BACKGROUND}) = -\log P_b\big(I(p)|p \in \text{background}\big)$$

$$V_p(\text{FOREGROUND}) = -\log P_f\big(I(p)|p \in \text{foreground}\big)$$

The probability density functions P_b, P_f can be modeled as two separate gaussian mixture models which have been learnt beforehand [25].

For the pairwise potential functions a contrast sensitive Potts model can be used as in the case of stereo matching.

Object detection and localization:

All examples shown so far concerned low-level vision problems. Markov

Random Fields, however, can be also used for modeling higher level problems. In the case of the object detection problem, we can model an object as a collection of parts with each part having a specific ideal appearance. We can also imagine that some of the parts are connected to each other by springs which try to keep them in an ideal relative distance [46]. We then want to find a configuration for the parts so that, on one hand, their appearance matches the underlying image data and, on the other hand, springs are not stretched too much (see Figure 1.8).

In this case the labels designate the position of the parts and correspond to all 2D positions in the image grid:

$$\mathcal{L} = \{1, 2, ..., w\} \times \{1, 2, ..., h\},$$

where w and h represent the image width and height.

The nodes V of the graph $\mathcal{G}$ correspond to the parts, while the edges E show which parts are connected to each other by a spring.

The data term $V_p(x_p)$ measures how well the ideal appearance of the part corresponding to node p matches the underlying image data at position x_p i.e. at the position where the part is assumed to be located. E.g. if the ideal appearance of a part at node p is represented by a patch, say IDEAL_p, we could then set $V_p(x_p)$ to be equal to the sum of square differences (SSD) between the patches IDEAL_p and I_p (where I_p denotes the patch of the input image at location x_p):

$$V_p(x_p) = \text{SSD}(\ \text{I}_p,\ \text{IDEAL}_p\ ).$$

The pairwise potential function $V_{pq}(x_p, x_q)$ measures the deformation of the spring connecting two parts, i.e. the deviation from the ideal relative distance d_{pq} between the parts corresponding to nodes p and q:

$$V_{pq}(x_p, x_q) = ||(x_p - x_q) - d_{pq}||^2.$$

Both the ideal appearances of parts $\{\text{IDEAL}_p\}_{p \in V}$ as well as their ideal rela-

Fig. 1.8: We can model a face as a graph consisting of 4 objects: nose, mouth and two eyes. Each object is assumed to have a specific appearance, while some of the objects are connected to each other with springs. These springs try to keep the objects at an ideal relative distance. For detecting faces, we then need to locate that graph inside the image, but without stressing the springs too much.

tive distances $\{d_{pq}\}_{(p,q)\in E}$ are assumed to have been estimated from training images during a learning stage that has taken place beforehand.

1.3 State-of-the-art optimization methods for discrete Markov Random Fields

Besides the issues of data representation and the choice of the objective function, another very important issue is that of the optimization algorithm that will be used for finding an optimal solution. A major concern here is the problem of local minima existing in any non-convex objective function. Depending on how local minima are handled, optimization techniques are divided into two general categories:

Local optimization techniques: These are all techniques which are capable of computing just a local minimum and are thus very sensitive to the initial estimate.

Global optimization techniques: Strictly speaking, these are all techniques which can provably find the exact global optimum regardless of their starting point. In practice, however, any optimization method which is capable of extracting a nearly optimal solution and is not sensitive to the initial estimate is also said, loosely speaking, to belong to this class. This more loose definition of a global optimization method will be assumed hereafter.

In the case of a discrete Markov Random Field, this issue of local minima becomes very important and can greatly affect the quality of the generated solutions. The reason for this is because, on one hand, the associated MRF energy functions are highly non-convex and, on the other hand, even for Markov Random Fields with very simple pairwise potential functions, e.g. the Potts function, their optimization turns out to be an NP-complete problem. This has slowed down the early adoption of the optimization approach for MRFs, since local optimization techniques (like ICM [16]) that were initially used, have proved to be very inefficient in practice.

Over the last years, however, there has been a resurged interest in the use of discrete MRFs. The primary reason for this has been the introduction of two powerful global optimization techniques: graph-cuts (the α-expansion algorithm) and loopy belief propagation. These two algorithms have achieved improved results over older techniques and have been used in a large variety of computer vision tasks. E.g., in stereo matching, the top contenders for the best stereo matching algorithm rely either on graph cuts or on belief propagation. Furthermore, papers related to these algorithms have won recent academic rewards [71, 81, 82] and it would be no exaggeration to say that nowadays almost all papers which are related to inference of discrete MRFs and have also appeared in any of the top computer vision conferences, such as ICCV, CVPR or ECCV, are exclusively using either one of these two techniques.

Graph-cuts and belief propagation are two optimization methods that rely on entirely different principles to achieve their goal. A more detailed analysis of these methods will be given in chapter 2 but, for now, it suffices to briefly sketch an intuitive description of how these two techniques work:

α-expansion min-cut: This is a greedy iterative algorithm, which works like a gradient descent approach. At each iteration it tries to find, among a large number of solutions, the one which brings the greatest decrease in the energy of the MRF. This procedure is repeated until no further improvement in the energy of the MRF can be achieved, in which case the algorithm terminates.

The important thing here is that the α-expansion algorithm reduces the problem of selecting the best solution at each iteration to a binary labeling

problem which, in turn, is reduced into the problem of finding the minimum cut in an appropriately constructed capacitated graph. This last reduction forms the core of the α-expansion method. We should note however that this reduction is valid only in the case where each pairwise potential function $V_{pq}(\cdot,\cdot)$ of the Markov Random Field is a metric distance i.e., roughly speaking, $V_{pq}(\cdot,\cdot)$ satisfies the triangle inequality. The α-expansion algorithm can thus be applied only to MRFs with metric pairwise potentials, in which case it is guaranteed that it can always produce a solution which is close to the optimal one.

loopy belief propagation: BP is an iterative algorithm which works by propagating local messages along the nodes of a Markov Random Field. At each iteration every node sends outgoing messages to all of its neighboring nodes as well as accepts incoming messages from each one of its neighboring nodes. Sending and receiving new messages for the next iteration is based solely on the existing set of messages at the current iteration. This procedure is therefore repeated until all the messages stabilize, i.e. they cannot change too much per iteration, in which case it is decided that the algorithm should terminate.

Based on the above description, it becomes obvious that the central concept, on which belief propagation relies its operation, is of course the *messages*. At each iteration, every node must send one message per label (i.e. $|\mathcal{L}|$ messages in total) to each one of its neighbors. Intuitively, the message sent from a node p to another node q about label x_q indicates how likely node p thinks that node q should be assigned that specific label. After stabilization of all messages, each node chooses that label which has the maximum support based on all incoming messages at the node. One could thus argue that BP operates in a distributed fashion: in order to select their labels, nodes work in a cooperative manner by sending votes to each other about which labels they should finally prefer.

1.4 Thesis contributions

There are three major issues when designing an MRF optimization algorithm:

Range of applicability: this refers to what kind of discrete MRFs the algorithm can be applied to. Of course, an algorithm's applicability depends on the kind of assumptions that it makes in order to ensure a correct operation. These assumptions usually affect the form of the MRF objective function that can be optimized. We would obviously like the algorithm to be able to handle as general MRF energy functions as possible. This will ensure that the energy function can indeed capture a wide range of computer vision problems. E.g. if the MRF energy function that has been chosen for modeling a specific problem is too simple, then it may very well be the case that the "correct" solution to the problem does not actually coincide with the global minimum of that function. Obviously, there is not much one can hope to gain from MRF optimization if, even this basic condition, does not hold true.

Optimality properties: this issue refers to what kind of guarantees the algorithm can make with respect to the optimality of the solutions it generates. Computing the exact global optimum is usually out of the question since, for the great majority of "interesting" MRFs, optimizing their energy turns out to be an NP-complete problem. However, the algorithm should at least provably generate solutions that are nearly optimal, i.e. close to the true (but unknown) global optimum.

Computational efficiency: from a practical point of view it is also critical that the algorithm exhibits a low computational cost (both in time and space). This should be true especially for low level vision problems, where one usually has to deal with very large MRFs.

With respect to the issues mentioned above, the two state-of-the-art algorithms, graph-cuts and belief propagation, exhibit a complementary behavior. For instance, one of the main limitations of the α-expansion min-cut algorithm is that it is applicable only to a restricted class of MRFs. In particular, as already mentioned, it can be applied only to MRFs with metric pairwise potentials. This assumption, however, is too restrictive and does not hold for many of the MRFs that one may wish to use in computer vision (besides, it would be too naive for one to believe that, among the great variety of tasks encountered in computer vision, all of them will satisfy this rather artificial assumption). But for those MRFs which

do satisfy the property of having metric potentials, the α-expansion algorithm can always guarantee that the cost of its solution will be within a known factor of the optimum.

On the other hand, loopy belief propagation imposes no restrictions on the MRF's pairwise potential functions and can handle Markov Random Fields with arbitrary energies. However, one of its biggest limitations is currently considered the fact that it exhibits a very high computational cost when the cardinality of the label set $\mathcal{L}$ is very large. This means that, in practice, loopy belief propagation cannot be applied to MRF problems with a huge number of labels, something which severely limits its applicability with respect to problems in computer vision.

The main contribution, presented during the first part of this thesis, concerns the introduction of novel MRF optimization techniques, which manage to extend both graph-cuts as well as belief propagation in significant ways and can also help these algorithms to overcome all of their limitations mentioned above. A detailed description of the related contributions for each one of the two algorithms will be provided in the next chapters of this thesis. However, for the purpose of giving a quick overview of these contributions to the reader, a very brief description of them is following next.

More specifically, with respect to graph-cuts, a new framework for designing approximation algorithms is proposed whose theoretical roots lie on duality of linear programming . Its main contributions are briefly the following:

- It can handle a very wide class of MRFs with both metric and non-metric energy functions.
- Most importantly, even for non-metric energies, the proposed algorithms have guaranteed optimality properties (i.e. worst-case suboptimality bounds).
- It includes the min-cut α-expansion method merely as a special case (for metric energies).
- Besides the worst-case bounds, the proposed algorithms also provide much smaller per-instance suboptimality bounds, which prove to be very tight in practice (i.e. very close to 1). This actually implies that the generated solutions are always almost optimal.

- Finally, the proposed framework is offering new insights into existing graph-cut techniques.

On the other hand, a novel optimization technique which carries two important extensions over standard belief propagation is also proposed in this thesis. These two extensions are:

- *priority-based message scheduling*
- and *dynamic label pruning*

Both of them work in cooperation to resolve what is currently considered the main limitation of belief propagation, i.e. its inefficiency to handle problems with a huge number of labels.

Furthermore, to verify and demonstrate the effectiveness of our Priority-BP algorithm, we apply it to the problem of image completion, which currently forms a very active research topic in computer vision. In fact, a part of this thesis's contribution also relates to this problem as well. More specifically, a new exemplar-based framework for image completion is proposed which makes the following contributions:

- It treats the tasks of image completion, texture synthesis and image inpainting in a unified way
- Contrary to existing greedy techniques, all of these tasks are posed in the form of a discrete optimization problem with a well defined global objective function.
- Our method is automatic (i.e. no user intervention required) and manages to avoid greedy patch assignments by maintaining (throughout its execution) many candidate source patches for each block of missing pixels. In this way, visual inconsistencies are prevented during the image completion process.

Finally, in order to conclude this introductory chapter, we would like to note that discrete MRFs are currently ubiquitous in computer vision. Due to this fact, the task of performing proper inference on these MRFs is currently considered of utmost importance and requires the existence of effective global optimization techniques, that will go beyond the limitations of existing methods. Based on

these observations, we therefore strongly believe that the contributions made in this thesis can be of broad interest and will find use in a large number of tasks in computer vision.

CHAPTER 2

Background on Markov Random Fields

The role of this chapter is to provide fundamentals of Markov Random Fields. In particular, it provides a brief but self-contained introduction to Markov Random Fields (which are now defined in a more rigorous manner) but, most importantly, it gives a Bayesian justification for the form of their energy function, which constitutes one of the main goals of this chapter. In addition, the two current state of the art MRF optimization techniques, α-expansion and loopy belief propagation, are briefly described and some basic concepts, related to these two algorithms, are introduced. Finally, the chapter ends with a short review of some other MRF optimization techniques, that have been also used in computer vision over the past years.

As far as the laws of mathematics refer to reality, they are not certain;
and as far as they are certain, they do not refer to reality.

—Albert Einstein (1879-1955)

2.1 Introduction

As already mentioned in the previous chapter, the form of the MRF objective function that will be minimized plays a very important role in the modeling of a computer vision task. Therefore, in order to put an optimization-based method on firm theoretical grounds, one must derive the associated energy function using some well established criteria, while he should avoid relying on an objective function that was derived in a heuristic manner. Based on the well known fact that uncertainties inevitably exist in almost any computer vision task, it is no surprise that principles from probability and statistics have been the perfect candidates for deriving such established criteria up to now. For instance, the Maximum

Likelihood criterion (used when only knowledge about the distribution of the observations is available) or the Maximum Entropy principle [41] (used when only prior information is available) are two such well known examples. In the most interesting case where we want to make use of both a likelihood as well as a prior, then the *maximum a posteriori* (MAP) criterion is the one which is used most often [54, 128]. However, for imposing the MAP criterion, one needs to find a way for defining the prior probability efficiently. Moreover, for this prior to be effective, it must be able to account for all contextual constraints between objects participating in the problem. As we shall see next, Markov Random Fields provide a principled way for performing exactly this task, i.e. specifying the form of the prior distribution in such a manner that spatial interactions between objects are taken into account. Most importantly, however, MRFs are also able to achieve this task in a way which is very efficient, i.e. by making use of local interactions only.

2.2 Basics of Markov Random Fields

Markov Random Fields are particular cases of what is currently known as undirected graphical models. For defining a discrete Markov Random Field we need a discrete set of labels $\mathcal{L}$ as well as a graph $\mathcal{G} = (V, E)$ consisting of N objects (i.e. $|V| = N$). A Markov Random Field is then formed by associating with each node $p \in V$ a random variable x_p, which can take values in the label set $\mathcal{L}$. The sample space (or configuration space) $\mathcal{X}^N$ is the set of all N-dimensional vectors $x = \{x_p | p \in V\}$ with components lying in $\mathcal{L}$ and the cardinality of that space is equal to $|\mathcal{X}^N| = |\mathcal{L}|^N$. Also, the edges E of the underlying graph $\mathcal{G}$ represent probabilistic dependencies between random variables. A Markov Random Field thus defines a probability distribution, which assigns to each vector x in the sample space a probability mass $p(x)$. However, not all distributions are allowed by a Markov Random Field. A valid distribution $p(x)$ should respect the probabilistic dependencies implied by the graph edges. More specifically, the following definition holds:

Definition 2.1. *The random variables $x = \{x_p\}_{p \in V}$ are said to form a Markov Random Field with underlying graph $\mathcal{G}$, if whenever the sets A and C are separated*

in the graph $\mathcal{G}$ by a set B then the random variables x_A, x_C are conditionally independent given the variables x_B, i.e.

$$p(x_A, x_C|x_B) = p(x_A|x_B)p(x_C|x_B).$$

In the above definition A, B, C represent arbitrary subsets of nodes in V while the notation x_A denotes all random variables corresponding to nodes included in the set A, i.e. $x_A = \{x_p : p \in A\}$. We also say that the set of nodes B separates the sets A and C, if for any path in the graph $\mathcal{G}$ starting from A and ending in C, that path necessarily passes through at least one node belonging to B (see Figure 2.1).

This definition of a Markov Random Field is actually a generalization of one dimensional Markov Processes. In fact, in the context of one dimensional Markov Processes, the above definition merely translates to the well known fact that the past and the future observations are conditionally independent given the present observations. Based on the above definition, we also see that the role of the graph $\mathcal{G}$ is to act as a kind of filter for the allowed distributions: only those distributions which manage to pass all the conditional independence tests implied by the graph $\mathcal{G}$ make up the family of MRF distributions. Also, another nice thing with MRF distributions is that conditional independency between nodes can be easily inspected simply by applying a graph theoretic operation to the underlying graph $\mathcal{G}$.

Furthermore, the seemingly complex set of conditional independencies implied by the above definition of a Markov Random Field actually turns out to be equivalent to a much smaller set of local conditional independencies. More specifically, the following equivalent definition of a Markov Random Field also holds true:

Definition 2.2. *The random variables $x = \{x_p\}_{p \in V}$ are said to form a Markov Random Field, if the following two conditions hold:*

- *$p(x)$ is strictly positive for all x*
- *$p(x_p|x_{\mathcal{N}_p}) = p(x_p|x_{V-p})$, where $\mathcal{N}_p$ represents all neighboring nodes of p in $\mathcal{G}$, while $V - p$ denotes all nodes of the graph $\mathcal{G}$ except for p.*

The first condition needs to be assumed for some technical reasons (which are usually satisfied in practice), while the second condition simply states the fact

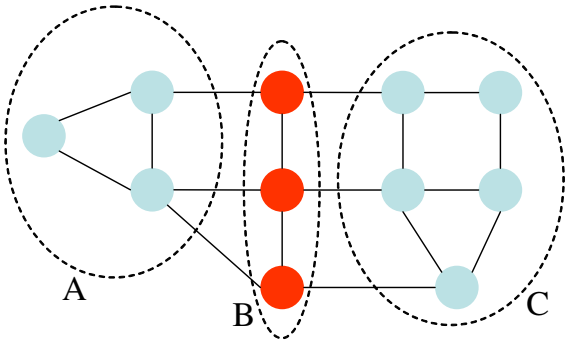


Fig. 2.1: The set of nodes in B separates the sets A and C since any path from A to C necessarily passes through B. Therefore, in any MRF with this graph, random variables x_A, x_C will be conditionally independent given variables x_B.

that any node in the graph $\mathcal{G}$ depends only on its immediate neighbors. This last condition is exactly what allows Markov Random Fields to model contextual constraints between objects in an efficient manner, since all contextual constraints are now enforced only through local interactions between neighboring nodes in $\mathcal{G}$, which constitutes a very important advantage of Markov Random Fields. In fact, this advantage is one of the main reasons why MRFs are so much preferred in practice.

2.3 Gibbs random fields and Markov-Gibbs equivalence

As already mentioned, any probability distribution that is defined on a Markov Random Field must satisfy certain constrains. But what is the exact form of such a distribution? To answer this question we must first define another concept.

Definition 2.3. *The set of random variables $x = \{x_p : p \in V\}$ is said to be a Gibbs random field with respect to the graph $\mathcal{G}$, if and only if its distribution $p(x)$ is a Gibbs distribution, i.e. it has the following form:*

$$p(x) = \frac{1}{Z} \cdot \exp\Big(-\sum_{C \in \mathbf{C}} V_C(x)\Big). \tag{2.1}$$

In the above definition, Z (which is also known as the *partition function*) is just a normalizing constant so that the sum of all probabilities is equal to unity. It is

thus defined as follows:

$$Z = \sum_{x \in \mathcal{X}^N} \exp\big(-\sum_{\mathcal{C} \in \mathbf{C}} V_{\mathcal{C}}(x)\big).$$

Also, the symbol $\mathbf{C}$ denotes the set of all *maximal cliques* in the graph $\mathcal{G}$. A clique is defined to be any fully connected subset of the nodes V of $\mathcal{G}$, while it is also called maximal, if it is not properly contained within any other clique. We should also note that whenever we mention the term clique hereafter, we will always implicitly refer to a maximal clique.

The symbols $V_{\mathcal{C}}(x)$ represent the so-called *clique potentials*, where each $V_{\mathcal{C}}(x)$ is a real function that depends only on the random variables contained in clique $\mathcal{C}$. Other than that, the clique potential functions are not restricted to have any specific form. Obviously the smaller the sum of all clique potentials for a certain sample x, the higher the probability mass $p(x)$, which will be assigned to that sample.

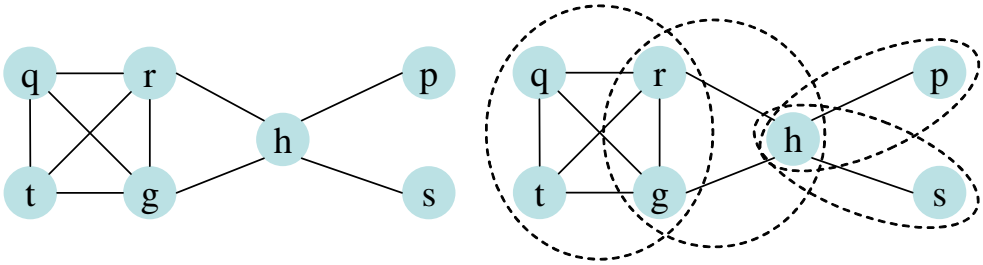
The following theorem [15,58] proves the equivalence between MRFs and Gibbs random fields:

Theorem 2.4 (Hammersley-Clifford). *A distribution $p(x)$ over a discrete random vector x is a Gibbs distribution (i.e. it is defined by equation (2.1)), if and only if the random variables x make up a Markov random field with respect to the graph $\mathcal{G}$.*

The practical value of this theorem is that it gives us an easy way to define the joint probability function of a Markov Random Field. In order to achieve that, it suffices for us to simply specify the clique potential functions. These functions therefore encode all desired contextual constraints between labels and, obviously, choosing the proper form for these functions constitutes the most important stage during MRF modeling.

Due to the fact that we will be dealing only with 1^{st} order Markov random fields throughout the rest of this thesis, the potential functions for all cliques with more than 2 elements will be assumed to be zero hereafter. Actually, the cliques of size 2 correspond to the edges E of the graph $\mathcal{G}$ and so for any such clique $\mathcal{C}$, say consisting of the elements p and q, its potential function will be denoted as:

$$V_{\mathcal{C}}(x) = V_{pq}(x_p, x_q).$$



(a) (b)

Fig. 2.2: The cliques of the graph in figure (a) are shown in figure (b). Therefore, any Gibbs distribution $p(x)$ on this graph can be expressed as: $p(x) \propto \exp\big(-V_{qrgt}(x_q, x_r, x_g, x_t) - V_{rhg}(x_r, x_h, x_g) - V_{hp}(x_h, x_p) - V_{hs}(x_h, x_s)\big)$

Therefore the joint probability distribution defined by the corresponding MRF will have the following form:

$$P(x) = \frac{1}{Z} \cdot \exp\Big(- \sum_{(p,q)\in E} V_{pq}(x_p, x_q)\Big). \tag{2.2}$$

2.4 *Maximum a posteriori* estimation

We now return to our initial objective, i.e. determining the actual form of the objective function $\mathcal{F}(\cdot)$ to be optimized. To this end, we recall here that in many applications we are given a set of noisy observations $d = \{d_p | p \in V\}$ and our goal is to estimate the true values of some hidden random variables $x = \{x_p | p \in V\}$. Many such applications were presented in the previous chapter. For instance, in the image restoration problem the observations d would correspond to a noisy version of an image, while x would represent the true underlying intensities of that image. Due to the existence of uncertainties in any application of this kind, a very common way to estimate the random variables x is by using the so-called *maximum a posteriori* criterion. From a Bayesian point of view, this criterion corresponds to minimizing the risk under the zero-one cost function. According to the MAP estimation, we choose to assign to the random variables x those values $\hat{x}$, which maximize the posterior distribution of x given all the observations d, i.e.:

$$\hat{x} = \arg\max_{x \in \mathcal{X}^N} p(x|d).$$

Using the Bayes rule, it follows that:

$$p(x|d) \propto p(d|x)p(x) \tag{2.3}$$

and so the posterior distribution turns out to be proportional to the product of the likelihood $p(d|x)$ and the prior $p(x)$. Therefore, in order to specify the posterior distribution, we first need to determine the form of $p(d|x)$ as well as $p(x)$.

Regarding the likelihood $p(d|x)$, one typically assumes that the observations d are conditionally independent given the hidden random variables x, which means that:

$$p(d|x) = \prod_{p \in V} p(d_p|x_p). \tag{2.4}$$

Without loss of generality we can express each conditional density $p(d_p|x_p)$ in exponential form, i.e.:

$$p(d_p|x_p) = \exp\big(-V_p(x_p)\big) \tag{2.5}$$

and so, due to the above two equations (2.4), (2.5), the likelihood function becomes equal to:

$$p(d|x) = \exp\big(-\sum_{p \in V} V_p(x_p)\big). \tag{2.6}$$

For fully determining the form of the posterior distribution, we are still left just with specifying the prior distribution $p(x)$. But this is exactly the point where Markov Random Fields come into play and prove to be extremely useful. In particular, we are going to assume that the hidden random variables x make up a Markov Random Field and so their prior distribution is going to be given by equation (2.2).

Therefore, by substituting the prior and the likelihood into the posterior (i.e. by substituting equations (2.2) and (2.6) into equation (2.3)), we finally get:

$$\hat{x} = \arg\max_{x \in \mathcal{X}^N} \exp\big(-\sum_{p \in V} V_p(x_p) - \sum_{(p,q) \in E} V_{pq}(x_p, x_q)\big), \tag{2.7}$$

which is obviously equivalent to minimizing the following function:

$$\mathcal{F}(x) = \sum_{p \in V} V_p(x_p) + \sum_{(p,q) \in E} V_{pq}(x_p, x_q). \tag{2.8}$$

This is, however, exactly the same objective function as the one presented in chapter 1 (see equation (1.1)).

2.5 State-of-the-art MRF optimization techniques

Given the Bayesian justification about the form of the objective function that was presented in the previous section, we now turn into the matter of how to optimize this function. For general MRFs this problem becomes NP-hard and so finding the optimal solution is far from being an easy task to achieve. During this section, we will briefly describe the main ideas behind the two currently dominating approaches for tackling this problem:

- α-expansion [27] and
- loopy belief propagation [103]

We should note that both of these methods are currently considered state-of-the-art with respect to the task of optimizing a Markov Random Field. Furthermore, they have already been applied to a variety of problems in computer vision.

2.5.1 The α-expansion algorithm

Before describing the algorithm, we very briefly review some basic concepts related to graph cuts. Let $\mathcal{G}_0 = (V_0, E_0)$ be a weighted graph. We assume that this graph has two distinguished vertices, the source and the sink,which are also called the terminals. Then any subset of edges $\mathcal{C} \subset E_0$ is called a *cut* if after removing all the edges in $\mathcal{C}$ then the two terminals are separated in the induced graph $\mathcal{G}_0(\mathcal{C}) = (V_0, E_0 - \mathcal{C})$. In addition, there should be no proper subset of $\mathcal{C}$ that could separate the two terminals. We also associate a certain cost to each of the cuts in $\mathcal{G}$, where the cost of a cut $\mathcal{C}$ is defined to be equal to the sum of all the weights on the edges of $\mathcal{C}$. The *minimum cut* problem then simply amounts to finding the cheapest cut among all cuts that separate the two terminal nodes (see Figure 2.3).

We should note that there exist standard combinatorial-based algorithms (of low order polynomial complexity) for solving the minimum cut problem. Most of

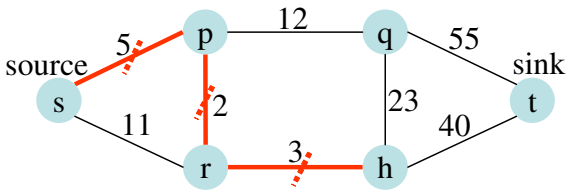


Fig. 2.3: A cut consisting of the edges sp, pr, rh is shown in this figure. This is, in fact, the minimum st-cut in the above graph since there is no other st-cut with cost less than 10.

them are actually based on solving another related problem, the max-flow problem, which can actually be shown to be equivalent to the problem of finding the minimum cut. According to the max-flow problem we seek to find the maximum amount of flow that can leave the source and arrive at the sink, while passing through any of the edges of the graph. In this case, the weight of an edge is interpreted as the edge's capacity, i.e. it represents the maximum flow that can pass through that edge. As the following well known theorem indicates, minimum cut and max-flow are equivalent to each other [55]:

Theorem 2.5 (max-flow min-cut). *The maximum amount of flow from the source to the sink equals the cost of the minimum cut that separates the source and the sink.*

Before describing the α-expansion method, we should also note that this algorithm can be applied only to MRFs for which each pairwise potential function $V_{pq}(\cdot,\cdot)$ is a metric, i.e. it satisfies the following properties for any triplet of labels a, b, c:

$$V_{pq}(a,b) \leq V_{pq}(a,c) + V_{pq}(c,b) \tag{2.9}$$

$$V_{pq}(a,b) = 0 \Leftrightarrow a = b \tag{2.10}$$

$$V_{pq}(a,b) = V_{pq}(b,a) \tag{2.11}$$

The α-expansion method is an iterative gradient descent algorithm which works in the following manner: At the start of each iteration a current solution x exists. Then a large set of candidate solutions is constructed by perturbing the current solution x. Among all these candidate solutions, the α-expansion algorithm tries to find the one which brings the greatest decrease in the MRF energy. The resulting solution x' then replaces x and becomes the new current solution. This

process is repeated until no more decrease in the MRF energy can be achieved. Pseudocode for the α-expansion method is shown in algorithm 1.

Algorithm 1 The α-expansion algorithm

```
1: start with an arbitrary labeling x
2: set success = 0
3: for each label α in 𝓛 do
4:    find x' = arg min 𝓕(x̃) among x̃ within one α-expansion of x
5:    if 𝓕(x') < 𝓕(x) then
6:       x = x'
7:       success = 1
8:    end if
9: end for
10:
11: if success = 1 then
12:    goto 2
13: else
14:    set x̂ = x
15: end if
16: return x̂
```

More specifically, given the current solution x and any label, say α, the algorithm tries to find the best solution among the set of all possible α-expansions of x (i.e. the set of all α-expansions of x corresponds to the set of candidate solutions mentioned above). During each iteration this is repeated for all labels α in $\mathcal{L}$. A solution x' is called an α-expansion of another solution x if and only if for all $p \in V$ it holds that either $x'_p = x_p$ or $x'_p = \alpha$ [27]. In other words, during the transition from x to x' any set of nodes are allowed to change their labels to α (see Figure 2.4).

The problem of finding the best α-expansion, say x', of the current solution x can be shown to be equivalent to optimizing a binary MRF problem with labels $\{\text{KEEP}, \text{CHANGE}\}$. Intuitively, assigning the label KEEP to a node p corresponds to $x'_p = x_p$, i.e. node p keeps its current label, while assigning the label CHANGE to a node p corresponds to $x'_p = \alpha$, i.e. node p changes its label into α. The contribution of the α-expansion algorithm is that it manages to show that the cost of any solution to this binary MRF problem actually corresponds to the cost of a cut in an appropriately constructed two-terminal graph whose capacities depend on the current solution x. Therefore, optimizing the binary MRF (or in other words extracting the best α-expansion move) reduces to finding the minimum cut in the capacitated graph. As the α-expansion algorithm continually tries to find

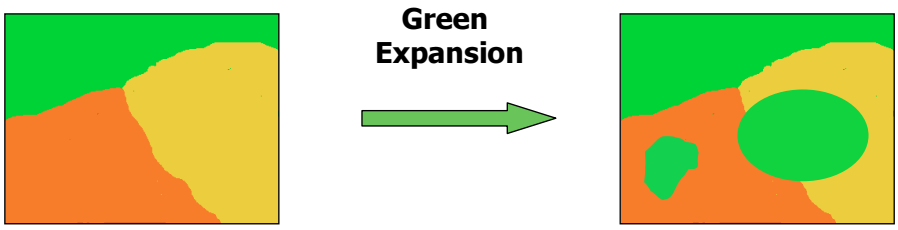


Fig. 2.4: The labeling shown on the right image is an example of a green-expansion of the labeling on shown on the left image. In this case, labels correspond to colors.

the best expansion moves, it becomes evident that this algorithm actually tries to optimize the original MRF by repeatedly estimating the minimum cut in a series of capacitated graphs.

The metric assumption for the pairwise potentials $V_{pq}(\cdot,\cdot)$ is necessary because otherwise the equivalence between finding the best expansion move and finding the minimum cut in the (appropriately constructed) capacitated graph does not hold true any more.

2.5.2 Loopy belief propagation

The α-expansion was a combinatorial-based algorithm. On the other hand, belief propagation is based on an entirely different principle: it tries to find a MAP estimate by iteratively solving a finite set of equations until a fixed point is obtained.

However, before one is able to understand how this set of equations comes up, one must first get acquainted with the notion of *"messages"*, which is another central concept in BP. In fact, belief propagation is nothing more than an iterative algorithm, which works by continuously propagating local messages between the nodes of the MRF graph. At every iteration, each node sends messages to all of its neighboring nodes, while it also accepts messages from these nodes. This process repeats until all messages stabilize, i.e. they do not change any more. Therefore, the set of equations whose fixed point one tries to find, actually corresponds to the equations governing how messages are updated during each iteration.

The set of messages sent from a node p to a neighboring node q will be denoted by $\{m_{pq}(x_q)\}_{x_q \in \mathcal{L}}$. Therefore, the total number of such messages is always $|\mathcal{L}|$ (i.e. there exists one message per label in $\mathcal{L}$). Intuitively, the meaning of the

message $m_{pq}(x_q)$ is that it expresses how likely node p thinks that node q should be assigned label x_q. Furthermore, whenever we say that node p sends a message $m_{pq}(x_q)$ to node q, what we practically mean is that the following update of the message $m_{pq}(x_q)$ is taking place:

$$m_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \Big\{ V_{pq}(x_p, x_q) + V_p(x_p) + \sum_{r:\ r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p) \Big\} \tag{2.12}$$

The interpretation of the above equation is that if node p wants to send the message $m_{pq}(x_q)$ to node q (i.e. if node p wants to tell how likely it thinks that label x_q should be assigned to node q), then node p must first traverse all of its own labels $x_p \in \mathcal{L}$ and then decide which one of them provides the greatest evidence for assigning label x_q to node q. But for taking this decision, node p must consider two factors:

- First, assuming that label x_p is assigned to node p, that node must consider how compatible label x_p is with label x_q (this is measured by the term $V_{pq}(x_p, x_q)$)

- And second, node p must also measure what is the likelihood that he is assigned label x_p.

  Obviously, on one hand, this likelihood will depend on the observed data at node p (see term $V_p(x_p)$). On the other hand, node p must also ask the opinion of its remaining neighbors about label x_p (this is measured by the sum $\sum_{r:r \neq q,(r,p) \in \mathcal{E}} m_{rp}(x_p)$). Therefore, before a node p sends a message to another node q, he must first consult the rest of its neighbors by receiving messages from them (see Figure 2.5). Put otherwise, during belief propagation, all MRF nodes work in cooperation in order to make a decision about the labels that they should finally choose. This cooperation between nodes is reflected by the exchange of opinions (i.e. messages), which is taking place during the algorithm.

The updating of messages, according to equation (2.12), continues until all of the messages have finally converged. Then, after convergence, a set of so-called *beliefs* $\{b_p(x_p)\}_{x_p \in \mathcal{L}}$ is computed for every node p in the MRF. These beliefs are

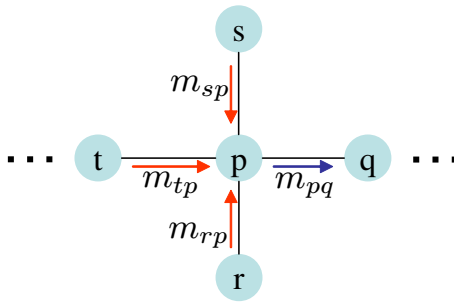


Fig. 2.5: If a node p wants to send a message $m_{pq}(x_q)$ to a neighboring node q, then it must make use of the messages $m_{sp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from the rest of its neighbors.

estimated using the following equation:

$$b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p)\in\mathcal{E}} m_{rp}(x_p). \tag{2.13}$$

The intuitive meaning of belief $b_p(x_p)$ is that it roughly expresses how likely node p thinks that label x_p should be assigned to him. Obviously, this will again depend on two things:

- on one hand, node p must take into account the observed data at that node (see term $V_p(x_p)$)
- on the other hand, node p must also consider the advice given by all of its neighbors about label x_p, which is accounted by the sum $\sum_{r:(r,p)\in\mathcal{E}} V_{rp}(x_p)$ in equation (2.13) (see also Figure 2.6).

Based on the above observations, once all beliefs have been computed, each node is then assigned the label having the maximum belief:

$$\hat{x}_p = \arg\max_{x_p\in\mathcal{L}} b_p(x_p). \tag{2.14}$$

Strictly speaking, beliefs actually approximate the so-called max-marginals i.e. each belief $b_p(x_p)$ approximates the maximum conditional probability that can be obtained given the fact that node p has been already assigned the label x_p[1].

[1]Actually, as we are working in the $-\log$ domain, beliefs approximate the opposite of the min-marginals e.g. $b_p(x_p)$ approximates the opposite of the minimum energy that can be obtained given that node p has already been assigned label x_p.

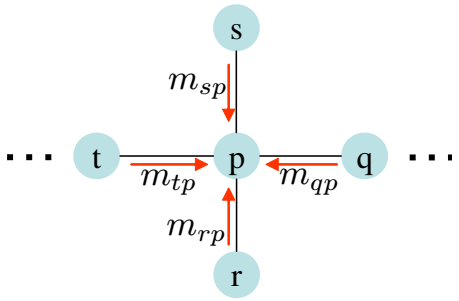


Fig. 2.6: If a node p wants to calculate its belief $b_p(x_p)$ about any of the labels $x_p \in \mathcal{L}$, it must then collect the messages $m_{sp}(x_p), m_{qp}(x_p), m_{rp}(x_p), m_{tp}(x_p)$ coming from all of its neighboring nodes.

Algorithm 2 Belief propagation algorithm

for all $(p, q) \in E$ **do** {initialize all messages to zero}
  for all labels $x_q \in \mathcal{L}$ **do**
    set $m^0_{pq}(x_q) = 0$
  end for
end for

for $k = 1$ to K **do** {update messages iteratively in K iterations}
  for all messages m^k_{pq} and all labels $x_q \in \mathcal{L}$ **do**
    $m^k_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \{V_{pq}(x_p, x_q) + V_p(x_p) + \sum_{r:r\neq q,(r,p)\in\mathcal{E}} m^{k-1}_{rp}(x_p)\}$
  end for
end for

for all nodes $p \in V$ **do** {compute beliefs}
  for all labels $x_p \in \mathcal{L}$ **do**
    $b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p)\in\mathcal{E}} m^K_{rp}(x_p)$
  end for
end for

for each node $p \in V$ **do** {assign the labels of maximum belief}
  set $\hat{x}_p = \arg\max_{x_p \in \mathcal{L}} b_p(x_p)$
end for

Pseudocode for the belief propagation method is shown in algorithm 2. It can be proved that, in the case of a tree structured graph, belief propagation is exact, i.e. the exact global optimum of the MRF energy is computed. Not only that, but it can be actually shown that this global optimum may be computed in just one iteration. However, if the MRF graph contains cycles then no such guarantee can be provided and belief propagation can only estimate a local optimum. Moreover, many iterations are then needed (this also explains why the algorithm has been given the name *loopy belief propagation* in this case). On the other hand, up to now, it has been experimentally proved that belief propagation can produce

very good results for a wide variety of computer vision tasks. Furthermore, there have also been some results, which indicate that the solutions generated by belief propagation possess certain optimality properties.

Finally, we should note that there are actually two versions of belief propagation: the max-product and the sum-product algorithm. The difference is that sum-product computes the marginal posterior of each node, while max-product maximizes the posterior of each node, i.e. it computes the max-marginals. The algorithm that has been presented above corresponds to the max-product algorithm but, due to the fact that we are using negative log probabilities, instead of maximizing products of terms we are actually minimizing sums of terms.

2.6 Other MRF optimization methods in vision

Except for the two specific state-of-the-art algorithms that have been presented in the previous section, there also exist some other methods related either to graph-cuts or to belief propagation. These methods will be reviewed separately in those chapters that are dealing specifically with graph-cuts and belief propagation.

For now, and before finishing this chapter, we would just like to mention some other techniques, besides graph-cuts and belief propagation, which also have been used in computer vision for the optimization of Markov Random Fields. Obviously, as the full list of MRF optimization techniques is vast, we will be able to refer only to a selected and very small subset of these methods.

To start with, it is worth mentioning the so-called Markov Chain Monte Carlo (MCMC) methods [8, 91, 110, 133, 134]. These are actually techniques which can be used for sampling any type of distribution. However, thanks to a procedure named *simulated annealing* [31, 74], sampling can be also used for optimization purposes. Simulated annealing is inspired by an analogy with metallurgy, in which slow cooling (annealing) is used to produce metal that is tougher than that which is produced from rapid cooling. When a Markov chain simulation (typically a Metropolis sampler [98]) is used to sample a Gibbs distribution, the analogous procedure is to introduce a temperature parameter T into the Gibbs distribution, which is gradually reduced from a very high initial value to a value close

to zero. When T is large the sampler can move freely through state space, thus escaping poor local minima. On the other hand, when the temperature is close to zero the distribution mass is concentrated in the area near the global minimum and so sampling becomes equivalent to optimization. The hope is that, as T is gradually decreased, one will be able to track the global minimum. Asymptotically, simulated annealing can always guarantee to extract the global optimum. This, however, requires infinite time in practice and so only local minima can be estimated.

Another class of techniques, resembling the way simulated annealing works, are the so-called *continuation methods* with graduated non convexity (GNC) [20] being one such example. In these methods the role of the temperature is played by another parameter γ. In this case, the intractable non-convex energy function is approximated by a sequence of energy functions parameterized by this parameter γ. When γ is large the energy function becomes strictly convex and so locating the global optimum is easy. However, as γ decreases, the energy function becomes non-convex and local minima are starting to appear. The hope is that if γ is decreased gradually, then by tracking the sequence of local minima we will be able to locate the global optimum at the end. Unfortunately, however, continuation methods cannot provide any optimality guarantees about their solutions, except for certain special cases.

Another commonly used technique is *dynamic programming* [4, 10]. This is a technique which can extract the exact global optimum when applied to Markov Random Fields which are one dimensional. Actually, dynamic programming can be considered just as a special case of belief propagation in the case where the underlying graph is reduced to a chain. Of course, the fact that it can be applied only to one-dimensional MRFs severely limits its applicability, especially for problems of early vision which typically require a 2D MRF.

Iterated Conditional Modes (ICM) [16] is another deterministic optimization technique which has been proposed by Besag. ICM tries to maximize the joint probability of an MRF by maximizing local conditional probabilities sequentially. More specifically, at each step of the algorithm, an MRF node is chosen and that node is then assigned the label which minimizes the energy of the MRF under the condition that all other nodes keep their labels fixed. This is repeated for all nodes

of the MRF until convergence i.e. until the energy cannot decrease further. The main disadvantage of ICM is that it is greedy and very sensitive to initialization. Therefore, unless a good initial estimate is given to ICM, it can easily get trapped to poor local minima. Highest Confidence First (HCF) is a deterministic algorithm which tries to improve ICM and has been proposed by Chou and Brown [35]. Its feature is that it processes the nodes of the MRF in a specific order. To this end, it introduces an uncommitted label and then uses a certain strategy for choosing which MRF node to "committ" next. Experimental results show that HCF is, on the whole, better than ICM with respect to the task of minimizing the energy of an MRF.

Another technique, which has also been widely used in image and vision community, is the *relaxation labeling* method [35, 66, 100, 111]. The idea of relaxation labeling is to replace the discrete labels with continuous ones that must lie on a high dimensional simplex. In this way, the problem of optimizing a discrete MRF is converted into a constrained continuous optimization problem, which can then be solved using standard gradient descent techniques.

Finally, we should note that another class of optimization techniques are the so-called *Genetic* algorithms [56, 64]. These are optimization methods which are inspired by the principle of natural evolution in the biological world and try to simulate the evolutionary process: in a population of individuals, those who possess the highest goodness-of-fit values are the ones who finally survive. Although genetic algorithms are general purpose optimization methods, their main disadvantage is that they are mostly based on heuristic procedures.

CHAPTER 3

Approximate Labeling via Graph Cuts Based on Linear Programming

During this chapter, a new framework is presented for both understanding and developing graph-cut based combinatorial algorithms suitable for the approximate optimization of a very wide class of MRFs that are frequently encountered in computer vision. The proposed framework utilizes tools from the duality theory of linear programming in order to provide an alternative and more general view of state-of-the-art techniques like the α-expansion algorithm which is included merely as a special case. Moreover, contrary to α-expansion, the derived algorithms generate solutions with guaranteed optimality properties for a much wider class of problems, e.g. even for MRFs with semimetric potentials. In addition, they are capable of providing per-instance suboptimality bounds in all occasions, including discrete Markov Random Fields with an arbitrary potential function. These bounds prove to be very tight in practice (i.e. very close to 1), which means that the resulting solutions are almost optimal. The effectiveness of our algorithms is demonstrated by presenting experimental results on a variety of low level vision tasks, such as stereo matching, image restoration, image completion and optical flow estimation, as well as on synthetic problems.

One's mind, once stretched by a new idea, never regains its original dimensions.

—Oliver Wendell Holmes (1809-1894)

3.1 Introduction

To present the material of this chapter in its full generality, we will adopt a very small change in notation and terminology. In particular, we are going to introduce the *Metric Labeling Problem* (or ML for short), which slightly generalizes pairwise Markov Random Fields. Like the case of Markov Random Fields, the Metric Labeling Problem (that has been first introduced by Kleinberg and Tardos [75]) can capture a broad range of discrete classification problems that arise in computer vision. According to that problem's definition, the task is to classify a set V of n objects by assigning to each object a label from a given discrete set L of labels. To this end, we are also given a weighted graph $G = (V, E, w)$, where the set of edges E represents the pairwise relationships between the objects while the weight w_{pq} of an edge pq represents the strength of the relationship between objects p, q. Each labeling of the objects in V is represented by a function $x : V \rightarrow L$, where x_p denotes the label assigned to object p. Furthermore, like Markov Random Fields, any such labeling is associated with a certain cost which can be decomposed into terms of 2 kinds:

- On one hand, for each $p \in V$ there is a *label cost* $c_{p,a} \geq 0$ for assigning label $a = x_p$ to p. Intuitively, the label costs express the likelihood of assigning labels to objects.

- On the other hand, for each pair of objects p, q that are related (i.e. connected by an edge in the graph G) there is a so-called *separation cost* for assigning labels $a = x_p, b = x_q$ to them. This separation cost is equal to $w_{pq} d_{ab}^{pq}$ where, as already mentioned, the edge weight w_{pq} represents the strength of the relationship between p, q, while d_{ab}^{pq} is a distance function between labels measuring how similar two labels are. The intuition behind this definition of the separation cost is that objects which are strongly related to each other should be assigned similar labels. This helps in preserving the spatial coherence of the final labeling. Also, notice that each edge pq can have its own unique distance d_{ab}^{pq}. Furthermore, in the original formulation of Metric Labeling, the distance d_{ab}^{pq} was assumed to be a metric, i.e. $d_{ab}^{pq} = 0 \Leftrightarrow a = b,\ d_{ab}^{pq} = d_{ba}^{pq} \geq 0,\ d_{ab}^{pq} \leq d_{ac}^{pq} + d_{cb}^{pq}$ but here we will relax this assumption.

Based on these definitions the total cost $\mathcal{F}(x)$ of a labeling x equals:

$$\mathcal{F}(x) = \sum\nolimits_{p \in V} c_{p,x_p} + \sum\nolimits_{(p,q) \in E} w_{pq} d^{pq}_{x_p x_q}$$

and the goal is to find a labeling with the minimum total cost.

It is obvious that the definition of the Metric Labeling problem is almost identical to the definition of a Markov Random Field. For instance, if we assume that all edge weights are set equal to unity (i.e. $w_{pq} = 1$), then simply by setting:

$$c_{p,a} = V_p(a) \quad \text{(label costs} \leftrightarrow \text{single node potentials)}$$
$$d^{pq}_{ab} = V_{pq}(a,b) \quad \text{(distance function} \leftrightarrow \text{pairwise potentials)}$$

one can easily see that, in this case, optimizing the cost in this Metric Labeling problem is completely equivalent to minimizing the energy of a discrete MRF whose pairwise potential function has been replaced by the distance function between labels [75]. For this reason, we will hereafter use the terms Metric Labeling and Markov Random Field, as well as the terms pairwise potential and distance function interchangeably. Also, in order to simplify notation, we will hereafter assume that all edges share a common distance function d_{ab}, i.e.:

$$d^{pq}_{ab} \equiv d_{ab}.$$

Therefore, in this case, the function describing the Metric Labeling cost becomes equal to:

$$\mathcal{F}(x) = \sum\nolimits_{p \in V} c_{p,x_p} + \sum\nolimits_{(p,q) \in E} w_{pq} d_{x_p x_q}.$$

Due to the direct connection between Metric Labeling and Markov Random Fields that was mentioned above, solving the Metric Labeling problem is (in general) NP-hard and therefore one can only hope for methods that provide approximate solutions. To this end two are the main classes of methods that have been proposed so far. On one hand, there exist those methods, which are based entirely on combinatorial optimization [27, 59, 69, 113, 138]. An advantage for the methods of this class is that they are efficient. In addition, they have been applied with great success to many problems in computer vision. However, up to now, they have been interpreted only as greedy local search techniques. On the other

hand, there exists a second class of techniques which are based on linear programming [5, 34, 75], i.e. they rely on an entirely different principle. The methods of this class are very general and possess good theoretical properties. However, their main drawback is that they impose an intolerable computational cost, which actually makes them impossible to use for almost all practical problems encountered in computer vision. The reason for this intolerable computational cost stems from the fact that all these methods need to formulate Metric Labeling as an equivalent linear integer program, which can grow very large (i.e. it can have a huge number of variables). This is always the case if the cardinality of the set V is large as well (i.e. like the great majority of problems arising in early vision). A direct consequence of this fact is that it then becomes extremely expensive to obtain a solution to the corresponding linear relaxation of this integer program. For better illustrating this point, we can consider the following integer program (introduced by Chekuri et al. in [34]) that provides an equivalent formulation of Metric Labeling:

$$\min \sum_{p\in V}\sum_{a\in L} c_{p,a}x_{p,a} + \sum_{(p,q)\in E} w_{pq} \sum_{a,b\in L} d_{ab}x_{pq,ab} \tag{3.1}$$

$$\text{s.t.} \sum\nolimits_a x_{p,a} = 1 \qquad \forall\ p \in V \tag{3.2}$$

$$\sum\nolimits_a x_{pq,ab} = x_{q,b} \qquad \forall\ b \in L,\ (p,q) \in E \tag{3.3}$$

$$\sum\nolimits_b x_{pq,ab} = x_{p,a} \qquad \forall\ a \in L,\ (p,q) \in E \tag{3.4}$$

$$x_{p,a},\ x_{pq,ab} \in \{0,1\} \quad \forall\ p \in V,\ (p,q) \in E,\ \ a,b \in L$$

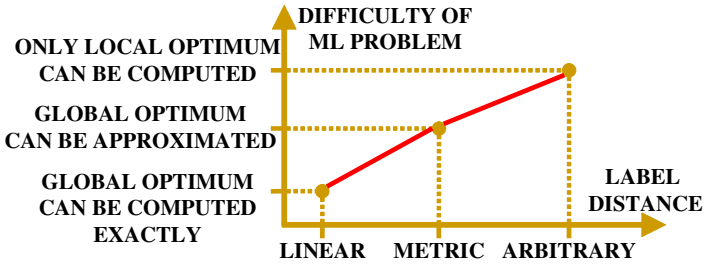
Here, for expressing the cost of Metric Labeling as a linear function, the variables $\{x_p\}_{p\in V}$ have been replaced with the binary variables $\{x_{p,a}\}_{p\in V,a\in L}$ and $\{x_{pq,ab}\}_{(p,q)\in E,a\in L,b\in L}$. The $\{0,1\}$-variable $x_{p,a}$ indicates that vertex p is assigned label a, while the $\{0,1\}$-variable $x_{pq,ab}$ indicates that vertex p is labeled a as well as vertex q is labeled b. The first constraints (3.2) simply express the fact that each vertex must receive exactly one label, while constraints (3.3), (3.4) maintain consistency between variables $x_{p,a}, x_{q,b}$ and $x_{pq,ab}$ in the sense that if $x_{p,a} = 1, x_{q,b} = 1$ they force $x_{pq,ab} = 1$ as well. Actually, the variables $x_{pq,ab}$, $x_{qp,ba}$ indicate exactly the same thing. Therefore, in order to reduce the number of variables and thus reduce the computational cost, we can safely eliminate one of these variables from the

integer program. To this end, we will hereafter assume without loss of generality that only one of $(p,q),\ (q,p)$ belongs to E for any neighbors p,q (this way, only one of the variables $x_{pq,ab}$, $x_{qp,ba}$ will appear in the definition of this integer program). Also, the notation "$p \sim q$" will hereafter denote the fact that p,q are neighbors to each other, i.e. "either only $(p,q) \in E$ or only $(q,p) \in E$". Nevertheless, despite the above mentioned reduction of variables, solving the linear relaxation of this integer program still remains an intractable task if the number of objects in V is not small enough. E.g. if we want to apply this method to a stereo matching problem, where the input images have size 512×512 while the maximum disparity is 64, then we would need to solve an LP with more than $512 \times 512 \times 64 \times 64 = 2^{30}$ variables. This certainly goes beyond the limits of any LP solver that is currently considered as state-of-the-art.

To overcome the limitations of current state-of-the-art methods a new framework is proposed in this document, which provides novel global minimization algorithms for the approximate optimization of the Metric Labeling problem (and thus of a very wide class of MRFs frequently encountered in computer vision). It makes use of the primal-dual schema of linear programming in order to derive efficient (i.e. combinatorial) approximation techniques with guaranteed optimality properties, thus bridging the gap between the two classes of approximation algorithms mentioned above. The major contributions of the proposed framework are the following:

1) It turns out that the difficulty of the Metric Labeling problem depends critically on the type of the chosen distance d_{ab} between labels (see Figure 3.1(a)). Up to now, one limitation of the state-of-the-art α-expansion method [27] was that it had to assume that this distance was a metric[1], i.e. it satisfied the triangle inequality. However this case often does not hold in practice, thus limiting the applicability of the α-expansion method. On the contrary, the algorithms derived in the proposed framework only require that the distance function is a semimetric, i.e. $d_{ab} = 0 \Leftrightarrow a = b,\ d_{ab} = d_{ba} \geq 0$. This is a much weaker assumption, since it essentially requires that the potential function of the associated MRF is just a symmetric and nonnegative function, two conditions which are usually satisfied

[1]We should note that it is possible for one to change the α-expansion method so that it can be applied to semimetrics as well [27]. However, this method of handling non-metric distances is very inefficient as well as inappropriate since it completely misses any structure of the distance function.



(a)

Method	Metric d_{ab}	Semimetric d_{ab}
α-β-swap	Local optimum	Local optimum
α-expansion	Approximation	×
our framework	Approximation	Approximation

(b)

Figure 3.1: (a) The difficulty of the ML problem depends critically on the type of label distance d_{ab} chosen. The global optimum in the case of a linear distance function can be found by using the technique described in [68], while an approximate solution in the case of a metric distance can be computed using the α-expansion method [27]. **(b)** A comparison of our framework with respect to existing state-of-the-art graph-cut methods.

in practice. In fact, the assumption of a symmetric distance is redundant, since none of the theorems in this chapter use it and so our algorithms can handle any distance for which $d_{ab} = 0 \Leftrightarrow a = b$, $d_{ab} \geq 0$. For this reason the term semimetric will hereafter imply just these conditions.[2]

This opens the way for applying our techniques to a much wider class of MRFs with much more general energy functions. Given that MRFs are ubiquitous in computer vision this also implies that these algorithms can handle many more instances of a large variety of computer vision tasks (including stereo matching, image restoration, image completion, optical flow estimation etc.). In all of these problems the use of more sophisticated MRF-based priors is allowed based on our framework, thus leading to a better modeling of the problem at hand. This is important since it is a well-known fact that the choice of the prior can play a very significant role in the quality of the generated solutions.

2) Furthermore, the quality of these solutions also depends critically on how close they are to the true optimum of the MRF energy function. A key contribution of our framework is that even in the case of a semimetric it can still guarantee that the generated solution will always be within a known factor of the global

[2]In fact our framework can be easily extended to even handle distances for which the condition $d_{ab} = 0 \Leftrightarrow a = b$ is not true.

optimum, i.e. a worst-case suboptimality bound can be provided in this case (see Figure 3.1(b)). This is in contrast to local optimization methods (e.g. ICM [16] or HCF [35]), which (by definition) can only guarantee to obtain just a local minimum (possibly far away from the true optimum).

3) In fact, in practice, the resulting solutions are much closer to the true optimum than what the worst-case approximation factors predict, i.e. they are nearly optimal. This can be verified thanks to our algorithms' ability of also providing per-instance suboptimality bounds which, in practice, prove to be much tighter (i.e. much closer to 1) than their worst-case counterparts. These bounds can therefore be used to access the optimality of the generated solutions and are, for this reason, very useful in deciding the ability of the chosen MRF for modeling the problem under consideration (e.g. the existence of a solution that is nearly optimal, but which does not look intuitively good implies that a different MRF should be chosen). Moreover, since these per-instance bounds are updated throughout the algorithm's execution they can be also used in assessing its convergence, thus possibly reducing the total running time. We should also note that the existence of such per-instance suboptimality bounds is a characteristic property of any primal-dual algorithm [34, 75, 140]. What is important to state in our case, however, is that on one hand these per-instance suboptimality bounds tend to be very tight in practice and on the other hand no large linear programs need to be solved for getting these bounds.

4) The generality and power of our framework is exhibited by presenting various algorithms, just one of which is proved to be equivalent to the α-expansion graph cut technique (i.e. a method which is currently considered state-of-the-art). Our framework therefore provides an alternative and more general view of these very successful graph-cut techniques, which can now be interpreted not merely as greedy local search but in terms of principles drawn from the theory of linear programming, thus shedding further light on their essence. This is an important advance which, we believe, may open the way for new related research and can thus lead to even better MRF optimization algorithms in the future. It also constitutes, in our opinion, one of the major contributions of the work described in this chapter. Moreover, the primal-dual schema, a powerful optimization tool that was already known to people in combinatorial optimization, is now also introduced to

the field of computer vision, which can prove to be a great benefit too.

The rest of the chapter is organized as follows. We review related work in section 3.2. In section 3.3 the primal-dual schema is presented, which will guide the design of all of our approximation algorithms. These algorithms are described in sections 3.4 - 3.6. More specifically, we will progressively present 3 different families of primal-dual algorithms, which are named PD1, PD2 and PD3 respectively. Algorithm PD1 forms the base for deriving and understanding the other two types of algorithms and so the main points of that algorithm are described thoroughly in section 3.4. In section 3.5 we derive PD2$_\mu$, which is the second family of primal-dual algorithms that are parameterized by a variable μ. Unlike algorithm PD1, all algorithms in this family can be applied only to metric MRFs. Furthermore, we show that the well-known α-expansion technique is equivalent to just one member of this family of algorithms. In particular, α-expansion arises if we simply set $\mu = 1$, i.e. it is equivalent to algorithm PD2$_{\mu=1}$. In section 3.6, we present algorithms PD3, which make up the third family of our primal-dual methods. These algorithms manage to extend, as well as generalize the α-expansion method (i.e. algorithm PD2$_{\mu=1}$) to the case of non-metric MRFs. In addition, despite this generalization, these algorithms manage to maintain the theoretical approximation guarantees of the PD2$_{\mu=1}$ algorithm. Experimental results are shown in section 3.7, while we conclude in section 3.8. We note that for reasons of clarity, all technical proofs for the theorems of this chapter are deferred to appendix A.

3.2 Related work

There is a vast amount of computer vision methods on how MRFs can be optimized. Such methods include for example the ICM-algorithm, the Highest-Confidence-First heuristic, multi-scale MRFs, relaxation labeling, graduated non-convexity and mean field annealing to mention just a few of them. However, all of the above-mentioned methods as well as the great majority of the methods in the literature are only able to provide a local minimum that can be arbitrarily far away from the true optimum, thus giving no guarantees about the quality of the resulting solutions (i.e. how close these are to the true optimum). Most closely related to our work are those (few) approaches that do provide such guarantees

about the optimality of their solutions.

One such class of approximation algorithms [5, 34, 75] is based on formulating the MRF optimization problem as a natural integer program. A linear programming relaxation of that integer program is then solved and a randomized rounding technique is being used to extract a near the optimum integer solution. Different authors choose different linear programs or rounding techniques to use for that purpose. Although these algorithms appear to have good theoretical properties, they are still impractical to use in problems of early vision since in that case the linear program to be solved becomes extremely large. Moreover, in order to provide any guarantees about the suboptimality of their solutions, they usually need to further assume that the MRF potential function is a metric.

Another class of approximation algorithms is based on combinatorial optimization. Out of these algorithms a very popular one is the α-expansion graph cut method [27, 138]. This can be interpreted as an iterative local search technique which, at each iteration, tries to extract a better solution (i.e. one with lower energy) by finding the minimum cut in a suitable graph. This state-of-the-art method has proved to be very efficient in practice and has been applied with great success to many problems in computer vision [24, 25, 51, 52, 73, 78, 80–82, 150]. Its drawback, however, is that (in its original formulation) α-expansion is only applicable to MRFs with a metric potential function. In fact, for some of these metrics, graph-cut techniques with better optimality properties seem to exist as well [59]. Recently, however, at the same time with our work, Rother et al. [112] managed to extend the α-expansion method to the semimetric case. In order to achieve this, they make use of a technique, which seems related to our PD3 algorithm.

Related to α-expansion is also the α-β-swap algorithm [27]. Although this is a more general method, as it applies to semimetric potentials as well, it does not seem to be as effective as α-expansion. This mainly has to do with the fact that it provides no guarantees about the optimality of its solutions and thus may very well get stuck to a bad local minimum. Finally, we should note that for a certain class of MRFs there also exist graph cut based methods, which are capable of extracting the exact global optimum [68, 69, 101, 113]. However these require the potential function to be convex, as well as the labels to be one-dimensional, a fact

which restricts their applicability.

For the sake of completeness we should also mention that there also exist those optimization algorithms that are based on belief propagation [103]. Although they impose no restrictions on the type of the MRF potential function to be chosen, one of their drawbacks is that they do not always converge. An exception to this is the recently proposed tree-reweighted belief propagation algorithm, which can be implemented in a way so that it provably converges [77]. Furthermore, another disadvantage of all Belief Propagation algorithms is that their theoretical optimality properties are not yet well understood, although progress has been made with respect to this issue over the last years [77, 77, 79, 140, 149].

3.3 The primal-dual schema

Let us consider the following pair of primal and dual linear programs:

$$\begin{array}{lll} \textsc{Primal}: & \min c^T x & \textsc{Dual}: \quad \max b^T y \\ & \text{s.t.} \quad Ax = b, x \geq 0 & \text{s.t.} \quad A^T y \leq c \end{array}$$

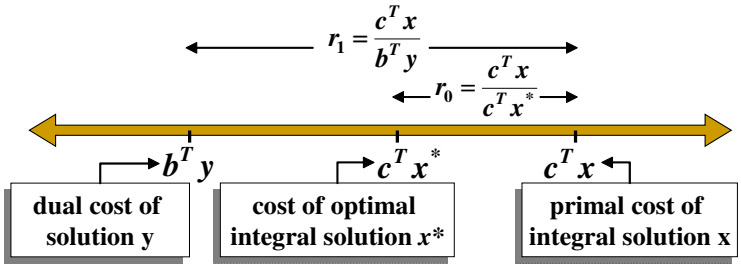
Here $A = [a_{ij}]$ represents an $m \times n$ matrix, while b, c are column vectors of size m, n respectively. We would like to find an optimal solution to the primal program under the additional constraint that its components are integer numbers. Due to this integrality requirement this problem is in general NP-complete and so we need to settle with estimating approximate solutions. A primal-dual f-approximation algorithm achieves that by use of the following principle:

Primal-Dual Principle. *If x and y are integral-primal and dual feasible solutions satisfying:*

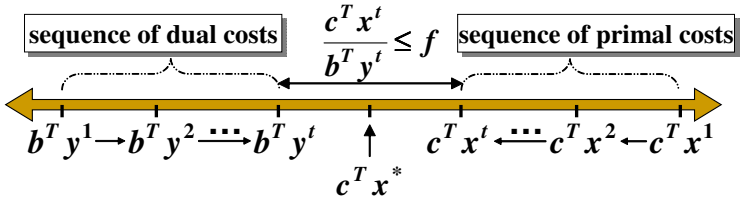
$$c^T x \leq f \cdot b^T y, \tag{3.5}$$

then x is an f-approximation to the optimal integral solution x^*, i.e. $c^T x \leq f \cdot c^T x^*$

This principle is essentially a direct consequence of the Weak Duality Theorem. The reason for this is rather simple and is also illustrated graphically in Figure 3.2(a): in particular, due to weak duality it will hold that the cost $c^T x^*$ of the optimal integral solution will always lie between the dual cost $b^T y$ and the primal cost $c^T x$. If we therefore manage to bring these two quantities, $b^T y$ and $c^T x$, close



(a) The primal-dual principle



(b) The primal-dual schema

Figure 3.2: (a) By weak duality the optimal cost $c^T x^*$ will lie between the costs $b^T y$ and $c^T x$ of any primal-dual pair of feasible solutions (x, y). Therefore if $b^T y$ and $c^T x$ are close enough (e.g. their ratio r_1 is $\leq f$) so are $c^T x^*$ and $c^T x$ (e.g. their ratio r_0 is $\leq f$ as well), thus proving that x is an f-approximation to x^*. **(b)** Dual and primal feasible solutions make local improvements to each other until the final costs $b^T y^t$, $c^T x^t$ are close enough (e.g. their ratio is $\leq f$). We can then apply the primal-dual principle and conclude that x^t is an f-approximation to x^*.

to each other (e.g. by making their ratio $r_1 = c^T x / b^T y$ less or equal to f) then we will have succeeded in making the costs $c^T x^*$ and $c^T x$ to come close to each other as well (e.g. their ratio $r_1 = c^T x / c^T x^*$ will also be less than f), thus proving that x is an f-approximation to x^*. The above principle lies at the heart of any primal-dual technique. In fact, the various primal-dual methods mostly differ in the way that they manage to estimate a pair (x, y) satisfying the fundamental inequality (3.5). One very common way for that (but not the only one) is by relaxing the so-called primal complementary slackness conditions [136]:

Theorem (Relaxed Complementary Slackness). *If the pair (x, y) of integral-primal and dual feasible solutions satisfies the so-called relaxed primal complementary slackness conditions:*

$$\forall\, x_j > 0 \Rightarrow \sum_{i=1}^{m} a_{ij} y_i \geq c_j / f_j$$

then (x, y) also satisfies the Primal-Dual Principle with $f = \max_j f_j$ and therefore x is an f-approximation to the optimal integral solution.

The relaxed complementary slackness conditions is nothing but a restatement of the fundamental inequality (3.5) after taking advantage of the fact that solutions x, y satisfy the feasibility conditions of the primal and dual program respectively. Based on the above theorem, during a primal-dual f-approximation algorithm the following iterative schema is usually being used:

Primal-Dual Schema. *Keep generating pairs of integral-primal, dual solutions $\{(x^k, y^k)\}_{k=1}^t$ until the elements x^t, y^t of the last pair are both feasible and satisfy the relaxed primal complementary slackness conditions.*

This schema is illustrated graphically in Figure 3.2(b). At each iteration, based just on the current dual feasible solution y^k, we perturb the current primal feasible solution x^k so that its primal cost $c^T x^k$ comes closer to the dual cost $b^T y^k$. This is also applied in reverse (i.e. y^k is perturbed as well) and a new primal-dual pair, say (x^{k+1}, y^{k+1}), is thus generated. This is repeated until the costs of the final primal-dual pair are close enough. The remarkable thing with this procedure is that the two processes (i.e. the primal and the dual) make local improvements to each other and yet they manage to achieve an approximately global objective at the end. *Also, it is worth mentioning that one can thus devise different approximation algorithms merely by specifying a different set of slackness conditions (i.e. different f_j) each time.*

3.3.1 The primal and dual LPs (Linear Programs) corresponding to Metric Labeling

For the case of Metric Labeling we will use integer program (3.1) as our primal linear program after first relaxing its $\{0, 1\}$-constraints to $x_{p,a} \geq 0,\ x_{pq,ab} \geq 0$. The

dual of the resulting LP then has the following form:

$$\begin{aligned}
\max\ & \sum\nolimits_p y_p \\
\text{s.t.}\ & y_p \leq ht^y_{p,a} && \forall p \in V,\ a \in L && (3.6)\\
& y_{pq,a} + y_{qp,b} \leq w_{pq} d_{ab} && \forall a,b \in L, (p,q) \in E && (3.7)\\
& \text{where: } ht^y_{p,a} \equiv c_{p,a} + \sum\nolimits_{q:q\sim p} y_{pq,a} && && (3.8)
\end{aligned}$$

To each vertex p, there corresponds one dual variable y_p. Also, to each edge $(p,q) \in E$ (and any label a), there correspond 2 dual variables $y_{pq,a}$ and $y_{qp,a}$. All the dual variables $y_{pq,a}, y_{qp,a}$ will be called *"balance variables"* hereafter and we will also say that $y_{pq,a}$ is the *conjugate balance variable* of $y_{qp,a}$ (and vice versa). The auxiliary variables $ht^y_{p,a}$ will be called *"height variables"*. The reason for this name, as well as for introducing these redundant variables will become clear in the sections that are following. For defining a dual solution, only the balance variables $y_{pq,a}$, as well as the y_p variables must be specified. The height variables $ht^y_{p,a}$ can then be computed by (3.8).

Throughout this chapter we will be considering only feasible $\{0,1\}$-primal solutions, i.e. the primal variables $\{x_{p,a}\}_{p\in V,a\in L}$ and $\{x_{pq,ab}\}_{(p,q)\in E,a\in L,b\in L}$ will always take values in the set $\{0,1\}$. It is then trivial to see that such solutions can be completely specified once we know what label has been assigned to each vertex. For this reason, instead of maintaining all primal variables $\{x_{p,a}\}_{p\in V,a\in L}$ and $\{x_{pq,ab}\}_{(p,q)\in E,a\in L,b\in L}$, we will hereafter adopt the usual notation according to which a primal solution x refers just to a set of labels $\{x_p\}_{p\in V}$, where x_p denotes the label assigned to vertex p. Under this notation, $x_{p,a} = 1$ is equivalent to $x_p = a$, while $x_{pq,ab} = 1$ implies $x_p = a, x_q = b$. Based on this observation, it is then not difficult to check that the relaxed slackness conditions related to the $x_{p,a}$ variables are reduced to:

$$y_p \geq c_{p,x_p}/f_1 + \sum\nolimits_{q:q\sim p} y_{pq,x_p}, \tag{3.9}$$

while the slackness conditions related to the $x_{pq,ab}$ variables reduce to:

$$x_p \neq x_q \Rightarrow y_{pq,x_p} + y_{qp,x_q} \geq w_{pq} d_{x_p x_q}/f_2 \tag{3.10}$$

$$x_p = x_q = a \Rightarrow y_{pq,a} + y_{qp,a} = 0 \tag{3.11}$$

where we consider the cases $a \neq b$ and $a = b$ separately.

Our objective will therefore be to find feasible solutions x, y satisfying the above conditions (3.9), (3.10) and (3.11) for specific values of f_1 and f_2. Conditions (3.11) simply say that conjugate balance variables are opposite to each other. For this reason we set by definition:

$$y_{qp,a} \equiv -y_{pq,a} \quad \forall\, (p,q) \in E,\ a \in L \tag{3.12}$$

and so we do not have to worry about conditions (3.11) hereafter.

3.3.2 An intuitive view of the dual variables

A way of viewing/visualizing the dual variables, that will prove useful when later designing our approximation algorithms, is the following: for each vertex p, we consider a separate copy of the complete set of labels L. It is then assumed that all of these labels are objects, which are located at certain heights relative to a common reference plane (see Figure 3.3). The height of any label a at vertex p is given by the dual variable $ht^y_{p,a}$. Expressions like "label a at p is below/above label b" imply $ht^y_{p,a} \leqslant ht^y_{p,b}$. The role of the balance variables is then to contribute to the increase or decrease of a vertex's height. In particular, due to (3.8), the height of label a at p can be altered only if at least one of the balance variables $\{y_{pq,a}\}_{q:q\sim p}$ is altered as well. In addition, due to the fact that conjugate balance variables are opposite to each other (see (3.12)), changes in the height of label a at p also affect the height of that label at a neighboring vertex. In Figure 3.3, for example, each time we increase the height of label c at p, say by increasing balance variable $y_{pq,c}$, the height of c at the neighboring vertex q is decreased by the same amount due to the decrease of the conjugate variable $y_{qp,c}$.

Before proceeding let us also define some terminology. Let x, y be any pair of integral-primal, dual solutions. We will call the label that x assigns to p (i.e. x_p) the *active label at* p. We will also refer to the height of the active label at p (i.e. ht^y_{p,x_p}) as merely *the height of* p. We will call the sum of the heights of all vertices the *"Approximate Primal Function"* (or APF for short), i.e. $APF^{x,y} = \sum_p ht^y_{p,x_p}$. This function's name comes from the fact that if x, y satisfy the relaxed slackness conditions then it is easy to prove that APF approximates the primal objective

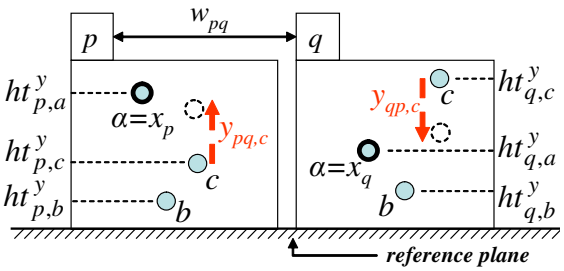


Fig. 3.3: Visualization of the dual variables for a graph G consisting of just 2 neighbors p, q while $L = \{a, b, c\}$. Each vertex holds a copy of all labels in L and all these labels are represented by circles which are located at certain heights specified by the ht variables. Label c at p is pulled up due to the increase of the balance variable $y_{pq,c}$ and so the corresponding label at neighboring vertex q is pulled down due to the decrease of the conjugate variable $y_{qp,c}$. The active labels of p, q are drawn with a thicker circle.

function.

Also, any balance variable in the set $\{y_{pq,x_p}\}_{q:q\sim p}$ (i.e. any balance variable of the form y_{pq,x_p} with q adjacent to p) will be called an *active balance variable at vertex* p. Another important quantity is the *load* between two neighbors p, q ($load^{x,y}_{pq}$) which is defined as the sum of the 2 active balance variables y_{pq,x_p}, y_{qp,x_q} i.e. $load^{x,y}_{pq} = y_{pq,x_p} + y_{qp,x_q}$. If relaxed slackness conditions (3.10) hold, then due to (3.10) and (3.7) it is easy to see that $w_{pq}d_{x_px_q}/f_2 \leq load^{x,y}_{pq} \leq w_{pq}d_{x_px_q}$ and so the load between p, q can be thought of as a *virtual separation cost* which approximates the actual separation cost $w_{pq}d_{x_px_q}$ between p, q.

3.3.3 Applying the primal-dual schema to Metric Labeling

Most of our primal-dual algorithms will achieve an approximation factor of

```
1: k ← 1; x^k ←INIT_PRIMALS( ); y^k ←INIT_DUALS( );
2: LabelChange ← 0
3: for each label c in L do
4:    ȳ^k ← PREEDIT_DUALS(c, x^k, y^k);
5:    [x^{k+1}, ȳ^{k+1}] ←UPDATE_DUALS_PRIMALS(c, x^k, ȳ^k);
6:    y^{k+1} ←POSTEDIT_DUALS(c, x^{k+1}, ȳ^{k+1});
7:    if x^{k+1} ≠ x^k then LabelChange ← 1
8:    k++;
9: end for
10: if LabelChange = 1 then goto 2;
11: if algorithm ≠ PD1 then y^fit ←DUAL_FIT(y^k);
```

Fig. 3.4: The basic structure of the algorithms PD1, PD2 and PD3.

$f_{app} = 2\frac{d_{max}}{d_{min}}$ where $d_{min} \equiv min_{a \neq b} d_{ab}$ and $d_{max} \equiv max_{a \neq b} d_{ab}$. Their basic structure can be seen in Figure 3.4. The initial primal-dual solutions are generated inside INIT_PRIMALS and INIT_DUALS. During an inner iteration (lines 4-8 in Figure 3.4) a label c is selected and a new primal-dual pair of solutions (x^{k+1}, y^{k+1}) is generated by updating the current pair (x^k, y^k). During this iteration, among all balance variables of y^k (i.e. $\{y^k_{pq,a}\}^{a \in L}_{p,q:p \sim q}$) only the balance variables of the c labels (i.e. $\{y^k_{pq,c}\}_{p,q:p \sim q}$) are modified. We call this a c-iteration of the algorithm. $|L|$ such iterations (one c-iteration for each label c in the set L) make up an outer iteration (lines 2-9 in Figure 3.4) and the algorithm terminates if no vertex changes its label during the current outer iteration.

During an inner iteration the main update of the primal and dual variables takes place inside UPDATE_DUALS_PRIMALS while PREEDIT_DUALS and POSTEDIT_DUALS modify the dual variables before and after the main update. The Primal-Dual algorithms that will be considered are named PD1, PD2, PD3. The DUAL_FIT routine, which is used only in the last two of them, serves only the purpose of applying a scaling operation to the last dual solution.

3.4 The PD1 algorithm

During this section we will assume that d_{ab} is a semimetric. In the case of the PD1 algorithm our goal will be to find feasible x, y satisfying slackness conditions (3.9), (3.10) with $f_1 = 1$ and $f_2 = f_{app}$. By replacing $f_1 = 1$ in (3.9) that condition becomes $y_p \geq ht^y_{p,x_p}$. Since it also holds that $y_p \leq \min_a ht^y_{p,a}$ (by the dual constraints (3.6)), it is easy to see that (3.9) reduces to the following 2 equations:

$$y_p = \min_a ht^y_{p,a} \tag{3.13}$$

$$ht^y_{p,x_p} = \min_a ht^y_{p,a} \tag{3.14}$$

In addition, $load^{x,y}_{pq} = y_{pq,x_p} + y_{qp,x_q}$ (by definition) and so by replacing $f_2 = f_{app}$ in (3.10) that condition reduces to:

$$x_p \neq x_q \Rightarrow load^{x,y}_{pq} \geq w_{pq} d_{x_p x_q} / f_{app} \tag{3.15}$$

Therefore the objective of PD1 is to find feasible x, y satisfying conditions (3.13)-(3.15) and for this, PD1 uses the following strategy: At each iteration it makes sure that conditions (3.13) and (3.15) are automatically satisfied by the current primal-dual pair. In addition, it makes sure that the current dual solution is feasible (primal solutions are always integral-feasible by construction). To this end it always imposes the following constraints to the current dual solution:

$$y_{pq,a} \leq w_{pq} d_{min}/2 \quad \forall\ a \in L,\ p \sim q \tag{3.16}$$

To see that (3.16) ensures feasibility, it is enough to observe that due to this constraint the following inequality can be derived: $y_{pq,a} + y_{qp,b} \leq 2w_{pq}d_{min}/2 = w_{pq}d_{min} \leq w_{pq}d_{ab}$ and so the dual constraints (3.7) hold true. This implies that solution y is indeed feasible, since the other dual constraints (3.6) already hold true due to condition (3.13).

All that remains then for PD1 to achieve its goal is just to ensure that after a finite number of iterations, slackness conditions (3.14) are satisfied as well. These conditions simply require that the active label of any vertex must be "lower" than all other labels at that vertex. So assuming that at the start of an outer iteration all conditions (3.13)-(3.16) except for (3.14) are satisfied, the update of the primal and dual variables roughly proceeds as follows:

DUAL VARIABLES UPDATE: Given the current primal solution (i.e. the current label assignment), we try to update the balance variables so that for each vertex its active label stays at the same height, while the rest of the labels at that vertex are raised above the active label. However, this may not be possible for all vertices because each time that one label goes up, another one at a neighboring vertex goes down. Furthermore, we cannot raise labels as much as we like since the balance variables cannot increase beyond the bounds imposed by constraints (3.16).

PRIMAL VARIABLES UPDATE: Therefore after the rearrangement of the labels' heights, there might still be some vertices violating condition (3.14), i.e. their active labels do not have the lowest height. We select a suitable subset of these vertices and assign to them new active labels with lower heights so that condition (3.14) is satisfied by as many of these vertices as possible. The reason we may not be able to do that for all the vertices is that we must still take care that conditions (3.15) are maintained as well.

However, by keep repeating this procedure, the number of vertices violating (3.14) decreases per iteration and so in the end all conditions (3.13)-(3.16) will hold true. Also note that it is always trivial to enforce conditions (3.13) (we simply set each dual variable y_p equal to $\min_a ht^y_{p,a}$).

The rearrangement of the labels' heights takes place in groups, one group per inner iteration. In particular, during a c-iteration only the heights of the c-labels are rearranged so that as many of these labels as possible are raised above the corresponding active labels. To this end solution y is changed into solution y' by changing only variables $\{y_{pq,c}\}_{p,q:p\sim q}$ (i.e. the balance variables of all c-labels) into $\{y'_{pq,c}\}_{p,q:p\sim q}$. The new heights will therefore be $ht^{y'}_{p,c}$. We must be careful, though, during this update of the c-heights. E.g. in Figure 3.5(a) we would like the c-labels at p and q to move at least as high as the active label $x_p{=}a$ of p and the active label $x_q{=}a$ of q respectively (label c at r does not need to move at all since it is already the active label of r). However, if we raise label c at p until it reaches x_p, say by increasing $y_{pq,c}$, then label c at q will go below x_q due to the decrease of conjugate variable $y_{qp,c}$, thus breaking condition (3.14) for q.

It turns out that the optimal update of the c-heights can be simulated by pushing the maximum amount of flow through an appropriate directed graph $G^{x,y}_c = (V^{x,y}_c, E^{x,y}_c, C^{x,y}_c)$, an example of which can be seen in Figure 3.5(b). The capacities $C^{x,y}_c$ of this graph depend on x, y (which are the primal-dual solutions at the start of the c-iteration), while its nodes $V^{x,y}_c$ consist of all the nodes of graph G (these are the *internal* nodes) plus two special *external* nodes the source s and the sink t. Furthermore, all nodes of $G^{x,y}_c$ are connected by two types of edges: *interior* and *exterior* edges (drawn with solid/dashed lines respectively in Figure 3.5(b)). All these edges are constructed as follows:

Interior edges: For each edge $(p, q) \in G$, we insert 2 directed interior edges pq and qp in graph $G^{x,y}_c$. The amount of flow f_{pq} coming out of p through pq will represent the increase of the balance variable $y_{pq,c}$, while the reverse flow f_{qp} will represent the decrease of the same variable $y_{pq,c}$. The total change of $y_{pq,c}$ will therefore be:

$$y'_{pq,c} = y_{pq,c} + f_{pq} - f_{qp} \tag{3.17}$$

The total change in $y_{qp,c}$ is defined symmetrically (since any flow coming out of p through pq will enter q and vice versa) and so $y'_{pq,c} = -y'_{qp,c}$, i.e. conjugate balance

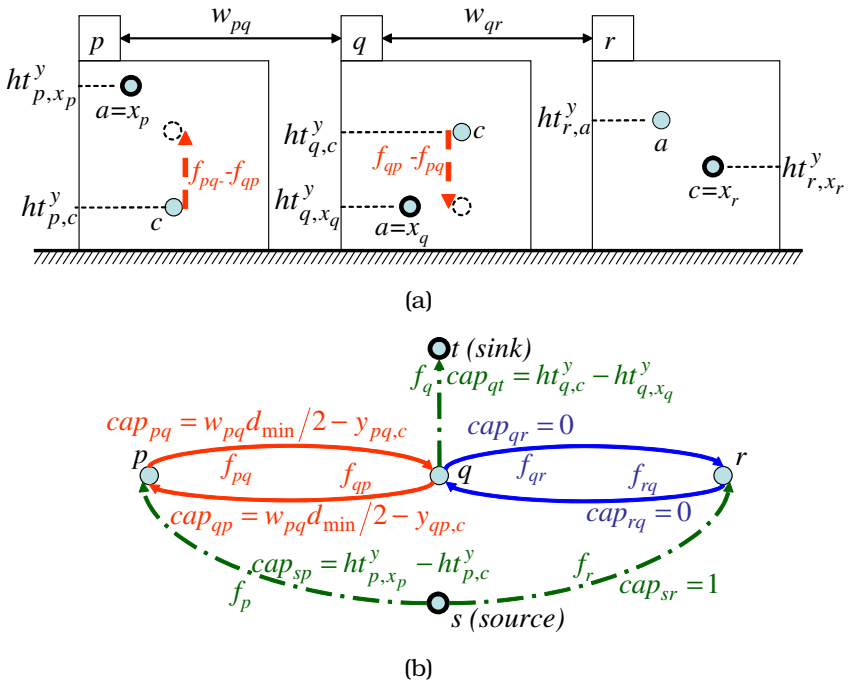


Fig. 3.5: (a) An arrangement of labels (represented by circles) for a graph G with 3 vertices p, q, r and 2 edges pq, qr of weights w_{pq}, w_{qr}. The label set is $L = \{a, c\}$. The thicker circles represent the active labels. Also, the red arrows indicate how the c labels will move in respond to an update of the balance variables while the dashed circles show the final position of those labels after the update. **(b)** The corresponding graph $G_c^{x,y}$ that will be used for updating the balance variables. Interior/exterior edges are drawn with solid/dashed lines respectively.

variables remain opposite to each other as they should.

Based on (3.17), it is obvious that the capacity cap_{pq} of edge pq will represent the maximum allowed increase of $y_{pq,c}$ (attained if $f_{pq} = cap_{pq}, f_{qp} = 0$), while a similar conclusion holds for cap_{qp}. Based on this observation cap_{pq}, cap_{qp} are assigned as follows: if the active label of p (or q) is equal to c then the height of c at p (or q) must stay fixed at this iteration and so we want no increase in $y_{pq,c}, y_{qp,c}$. Therefore, it should hold that:

$$x_p = c \;\text{ or }\; x_q = c \Rightarrow cap_{pq} = cap_{qp} = 0 \tag{3.18}$$

In all other cases (i.e. if $x_p \neq c, x_q \neq c$) cap_{pq}, cap_{qp} are set so that the values of the new balance variables $y'_{pq,c}$, $y'_{qp,c}$ can never exceed $w_{pq}d_{min}/2$ and feasibility conditions (3.16) are therefore maintained for the new dual solution y'. For this

reason cap_{pq}, cap_{qp} are set so that:

$$y_{pq,c} + cap_{pq} = w_{pq} d_{min}/2 = y_{qp,c} + cap_{qp}. \tag{3.19}$$

The capacities of blue/red edges in Figure 3.5(b), for example, are defined by (3.18)/(3.19) respectively.

Exterior edges: Each internal node p connects to either the source node s or the sink node t (but not to both of them) through an exterior edge. We have 3 possible cases to consider:

Case 1: If c is "below" x_p (i.e. $ht^y_{p,c} < ht^y_{p,x_p}$), then we would like to raise label c by exactly as much as needed so that it reaches label x_p. E.g. in Fig. 3.5(a) we would like that label c at p reaches label a at p. To this end, we connect the source node s to node p through a directed edge sp. The flow f_p passing through that edge will then represent the total increase in the height of label c, i.e.:[3]

$$ht^{y'}_{p,c} = ht^y_{p,c} + f_p \tag{3.20}$$

Therefore, based on (3.20), the capacity cap_{sp} of the edge sp will represent the maximum allowed raise in the height of c. Since we need to raise c only as high as the current active label of p, but not higher than that, we therefore set:

$$cap_{sp} = ht^y_{p,x_p} - ht^y_{p,c}. \tag{3.21}$$

The capacity of edge sp in Fig. 3.5(b) is defined this way.

Case 2: If c is not "below" x_p (i.e. $ht^y_{p,c} \geq ht^y_{p,x_p}$) and is also not the active label of p (i.e. $c \neq x_p$), then we can afford a decrease in the height of c as long as c remains "above" x_p (e.g. this is the case with label c at q in Fig. 3.5(a)). To this end, we connect p to the sink node t through directed edge pt. This time the flow f_p through edge pt will equal the total decrease in the height of c, i.e.:

$$ht^{y'}_{p,c} = ht^y_{p,c} - f_p \tag{3.22}$$

[3] To verify (3.20) it suffices to combine (3.17) with the flow conservation at node p which reduces to $f_p = \sum_{q:q\sim p}(f_{pq} - f_{qp})$. It then holds: $ht^y_{p,c} + f_p \overset{(3.8)}{=} \left(c_{p,c} + \sum_{q:q\sim p} y_{pq,c}\right) + f_p = \left(c_{p,c} + \sum_{q:q\sim p} y_{pq,c}\right) + \sum_{q:q\sim p}(f_{pq} - f_{qp}) \overset{(3.17)}{=} \left(c_{p,c} + \sum_{q:q\sim p} y_{pq,c}\right) + \sum_{q:q\sim p}(y'_{pq,c} - y_{pq,c}) = c_{p,c} + \sum_{q:q\sim p} y'_{pq,c} \overset{(3.8)}{=} ht^{y'}_{p,c}$

and so cap_{pt} will represent the maximum value of such a decrease. Therefore, based on the fact that c has to remain above x_p, we set:

$$cap_{pt} = ht^y_{p,c} - ht^y_{p,x_p}. \tag{3.23}$$

See edge qt in Fig. 3.5(b) for an example belonging to this case.

Case 3: Finally, if c is the active label of p (i.e. $c = x_p$) then we want to keep the height of c fixed at the current iteration (e.g. this is the case with label c at r in Fig. 3.5(a)). As in case 1, we again connect the source node s to node p through directed edge sp. This time, however, no flow passes through any interior edge incident to p (due to (3.18)). So $f_p = 0$ (due to the flow conservation at p) and the height of label c will not change (see (3.20)), as was intended. By convention we set the capacity cap_{sp} of edge sp equal to one in this case (see edge sr in Fig. 3.5(b)):

$$cap_{sp} = 1. \tag{3.24}$$

3.4.1 Update of the primal and dual variables

We are now ready to describe how the primal and dual variables are updated during a c-iteration, i.e. what actions are performed by each of the main routines of PD1.

PREEDIT_DUALS(c, x^k, y^k): For all of the considered algorithms this routine's role is to edit current solution y^k into solution $\bar{y}^k$ that will be used (along with x^k) as input for the construction of the graph $G_c^{x^k,\bar{y}^k}$ of the previous section. No editing is needed in the case of PD1 and so $\bar{y}^k = y^k$.

UPDATE_DUALS_PRIMALS$(c, x^k, \bar{y}^k)$: The primal-dual pair $x^{k+1}, \bar{y}^{k+1}$ is generated inside this routine. For the generation of $\bar{y}^{k+1}$, the graph $G_c^{x^k,\bar{y}^k}$ is constructed and a maximum flow algorithm is applied to it. The resulting flows are used in updating only the balance variables of the c labels as explained in the previous section (see (3.17)), i.e.:

$$\bar{y}^{k+1}_{pq,c} = \bar{y}^k_{pq,c} + f_{pq} - f_{qp}. \tag{3.25}$$

Therefore, due to (3.20), (3.22), the heights of all c labels will also change as:

$$ht_{p,c}^{\bar{y}^{k+1}} = ht_{p,c}^{\bar{y}^{k}} + \begin{cases} f_p \text{ if } p \text{ is connected to node } s \\ -f_p \text{ if } p \text{ is connected to node } t \end{cases} \tag{3.26}$$

Based on the new heights (e.g. see Fig. 3.6(b)), we now need to update x^k into x^{k+1} i.e. assign new labels to vertices. Since only the c labels have changed their heights (and thus only them may have gone above or below the active labels), this amounts to deciding whether a vertex keeps its current active label or is assigned the label c. On one hand, this should be based on whether the c-label of a vertex managed to "raise" as high as the active label of that vertex or is "below" it. On the other hand, we must also ensure that conditions (3.15) will still hold true for $x^{k+1}, \bar{y}^{k+1}$. It turns out that both criteria can be achieved just by considering the flows in $G_c^{x^k,\bar{y}^k}$ and making use of the following rule:

REASSIGN RULE. *Label c will be the new label of p (i.e. $x_p^{k+1} = c$), if and only if there exists an unsaturated*[4] *path between the source node s and node p. In all other cases, p keeps its current label i.e. $x_p^{k+1} = x_p^k$.*

See Fig. 3.6(c) for an example of applying the reassign rule. Based on this rule the following properties can then be shown to hold true for the resulting solutions $x^{k+1}, \bar{y}^{k+1}$ (see appendix A for a proof):

Lemma 3.1. *Any pair of primal-dual solutions $(x^{k+1}, \bar{y}^{k+1})$ satisfies the following properties:*

Property 1: *If at least one vertex has changed its active label, then it holds true that:*

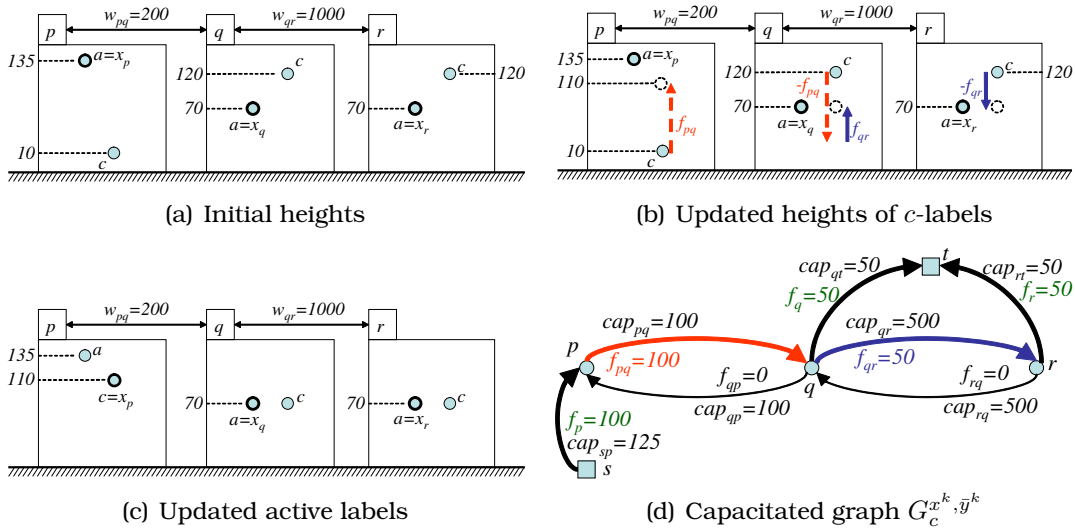
$$APF^{x^{k+1},\bar{y}^{k+1}} < APF^{x^k,\bar{y}^k}$$

Property 2: $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} \leq ht_{p,c}^{\bar{y}^{k+1}}$

Property 3: *If* $c = x_p^{k+1} \neq x_q^{k+1}$, *then* $\bar{y}_{pq,c}^{k+1} = \bar{y}_{pq,c}^{k} + cap_{pq}$

Property 1 can be used to prove that the algorithm will finally terminate (or else the APF function should decrease for ever, which obviously cannot be true assuming

[4] *A path is unsaturated if $flow < capacity$ for all forward arcs and $flow > 0$ for all backward arcs*



(a) Initial heights

(b) Updated heights of c-labels

(c) Updated active labels

(d) Capacitated graph $G_c^{x^k,\bar{y}^k}$

Fig. 3.6: (a) An initial arrangement of labels' heights at the start of a c-iteration. All vertices p, q, r are currently assigned label a (as indicated by the thick circles). **(b)** The new heights as updated by the UPDATE_DUALS_PRIMALS routine after applying a maximum flow algorithm to the graph in (d). Red and blue arrows show how the c-labels move due to the update of the balance variables. Movements due to changes in conjugate balance variables are drawn with the same line style and color. The dashed circles indicate the final positions of the c-labels. **(c)** The new active labels (thick circles) that were selected based on the "reassign rule". Only vertex p had to change its active label into c. This is so because only p still has its label c below its previous active label a or equivalently only edge sp of the graph in (d) is unsaturated while any paths to q or r are not. **(d)** The associated capacitated graph (assuming that all balance variables are initially zero) and the resulting flows due to the maximum flow algorithm. Notice that, as expected, the flows f_p, f_q, f_r at exterior edges reflect the total movement of the c-labels at p, q, r respectively. In this example the Potts metric has been used as distance d_{ab} (i.e. $a \neq b \Rightarrow d_{ab} = 1$).

integer capacities), while property 2 asserts that for any vertex its new active label is always "below" its c-label, as intended. Furthermore, property 3 can be used to prove that conditions (3.15) remain true for the current iteration.

POSTEDIT_DUALS$(c, x^{k+1}, \bar{y}^{k+1})$: However, for maintaining conditions (3.15) during the next iterations as well, it turns out that POSTEDIT_DUALS needs to change solution $\bar{y}^{k+1}$ into y^{k+1} so that all active balance variables of $\bar{y}^{k+1}$ become nonnegative variables of y^{k+1}, while also neither the *APF* nor any of the *loads* are altered during this change. In the case of PD1 it can be shown that any of the active balance variables of $\bar{y}^{k+1}$, i.e. any variable of the form $\bar{y}^{k+1}_{pq,x_p^{k+1}}$, $\bar{y}^{k+1}_{qp,x_q^{k+1}}$, may be negative during a c-iteration only if $x_p^{k+1} = x_q^{k+1} = c$. In this case POSTEDIT_DUALS simply needs to set $y^{k+1}_{pq,c} = y^{k+1}_{qp,c} = 0$ with no other differences existing between $\bar{y}^{k+1}, y^{k+1}$.

PD1's pseudocode is shown in Figure 3.7. Based on this definition the following

theorem can be proved asserting that PD1 always leads to an f_{app}-approximate solution:

Theorem 3.2. *The final primal-dual solutions generated by PD1 satisfy all conditions (3.13)-(3.16) and so they also satisfy the relaxed slackness conditions with* $f_1 = 1$, $f_2 = f_{app}$.

3.5 The PD2 algorithm

Algorithm PD2 (unlike PD1) can be applied only if d_{ab} is a metric. In fact, PD2 represents a family of algorithms parameterized by a variable $\mu \in [\frac{1}{f_{app}},\ 1]$. PD2$_\mu$ will achieve slackness conditions (3.9), (3.10) with $f_1 = \mu f_{app}$ and $f_2 = f_{app}$. The reason for $\mu \geq \frac{1}{f_{app}}$ is because $f_1 < 1$ can never hold.

A main difference between algorithms PD1 and PD2$_\mu$ is that the former is always generating a feasible dual solution at any of its inner iterations, while the latter will allow any of these dual solutions to become infeasible. However, PD2$_\mu$ ensures that the (probably) infeasible dual solutions are *"not too far away from feasibility"*. This practically means that if these solutions are divided by a suitable factor, they will become feasible again. This method (i.e. turning an infeasible dual solution into a feasible one by division) is also known as *"dual-fitting"* [136] in the linear programming literature.

More specifically, the PD2$_\mu$ algorithm generates a series of intermediate pairs of primal-dual solutions with the following properties: all of them satisfy slackness condition (3.10) as an equality with $f_2 = \frac{1}{\mu}$, i.e.:

$$x_p \neq x_q \Rightarrow load_{pq}^{x,y} = \mu w_{pq} d_{x_p x_q}. \tag{3.27}$$

In addition, the last one of these intermediate pairs satisfies the exact (i.e. $f_1 = 1$) slackness condition (3.9) which, as explained in section 3.4, reduces to:

$$y_p = \min_a ht_{p,a}^y \tag{3.28}$$

$$ht_{p,x_p}^y = \min_a ht_{p,a}^y \tag{3.29}$$

However, the dual solution of this last pair may be infeasible, since (although it

INIT_PRIMALS**:** initialize x^k by a random label assignment

INIT_DUALS
  $y^k = 0$
  for each pair $(p,q) \in E$ with $x_p^k \neq x_q^k$ **do**
    $y^k_{pq,x_p^k} = -y^k_{qp,x_p^k} = w_{pq}d_{min}/2 = y^k_{qp,x_q^k} = -y^k_{pq,x_q^k}$
  end for
  set $y_p^k = \min_a ht_{p,a}^{y^k}$ for all $p \in V$ ***{imposes (3.13)}***

{this routine generates $\bar{y}^k$***}***

PREEDIT_DUALS(c, x^k, y^k)**:** $\bar{y}^k = y^k$

{this routine generates $x^{k+1}, \bar{y}^{k+1}$***}***

UPDATE_DUALS_PRIMALS$(c, x^k, \bar{y}^k)$
  $x^{k+1} = x^k,\ \bar{y}^{k+1} = \bar{y}^k$
  Apply max-flow to $G_c^{x^k,\bar{y}^k}$ and compute flows f_p, f_{pq}
  set $\bar{y}^{k+1}_{pq,c} = \bar{y}^k_{pq,c} + f_{pq} - f_{qp}$ for all p, q with $(p,q) \in E$
  for all $p \in V$, set $x_p^{k+1} = c \Leftrightarrow \exists$ unsaturated path $s \rightsquigarrow p$ in $G_c^{x^k,\bar{y}^k}$

{this routine generates y^{k+1}***}***

POSTEDIT_DUALS$(c, x^{k+1}, \bar{y}^{k+1})$
  $y^{k+1} = \bar{y}^{k+1}$
  for each pair $(p,q) \in E$ with $x_p^{k+1} = x_q^{k+1} = c$ **do**
    if $\bar{y}^{k+1}_{pq,c} < 0$ **or** $\bar{y}^{k+1}_{qp,c} < 0$ **then** $y^{k+1}_{pq,c} = y^{k+1}_{qp,c} = 0$
  end for
  set $y_p^{k+1} = \min_a ht_{p,a}^{y^{k+1}}$ for all $p \in V$ ***{imposes (3.13)}***

Fig. 3.7: Pseudocode for the PD1 algorithm.

satisfies dual constraints (3.6) due to (3.28)) in place of constraints (3.7) it can be shown to satisfy only:

$$y_{pq,a} + y_{qp,b} \leq 2\mu w_{pq} d_{max} \quad \forall a, b \in L,\ (p,q) \in E \tag{3.30}$$

Nevertheless these conditions ensure that the last dual solution, say y, is not "too far away from feasibility". This means that by replacing y with $y^{fit} = \frac{y}{\mu f_{app}}$ we can then show that:

$$y^{fit}_{pq,a} + y^{fit}_{qp,b} = \frac{y_{pq,a} + y_{qp,b}}{\mu f_{app}} \overset{(3.30)}{\leq} \frac{2\mu w_{pq} d_{max}}{\mu f_{app}} = \frac{2\mu w_{pq} d_{max}}{\mu 2 d_{max}/d_{min}} = w_{pq} d_{min} \leq w_{pq} d_{ab}$$

meaning that y^{fit} is feasible. The generation of y^{fit} (given y) is exactly what the DUAL_FIT routine does. Furthermore, by using (3.27) and (3.30), it then takes

INIT_PRIMALS: initialize x^k at random

INIT_DUALS
  $y^k = 0$
  for all $(p,q) \in E$ with $x_p^k \neq x_q^k$ **do** ***{imposes (3.27)}***
    $y^k_{pq,x_p^k} = -y^k_{qp,x_p^k} = \mu w_{pq} d_{x_p^k x_q^k}/2$
    $y^k_{qp,x_q^k} = -y^k_{pq,x_q^k} = \mu w_{pq} d_{x_p^k x_q^k}/2$
  end for
  set $y_p^k = \min_a ht^{y^k}_{p,a}$ for all $p \in V$ ***{imposes (3.28)}***

PREEDIT_DUALS(c, x^k, y^k)
  $\bar{y}^k = y^k$
  for each $(p,q) \in E$ with $x_p^k \neq c, x_q^k \neq c$ **do**
    $a = x_p^k,\ b = x_q^k$
    $\bar{y}^k_{qp,c} = \mu w_{pq} d_{ac} - \bar{y}^k_{pq,a};\ \ \bar{y}^k_{pq,c} = -\bar{y}^k_{qp,c}$
  end for

POSTEDIT_DUALS$(c, x^{k+1}, \bar{y}^{k+1})$
  edit $\bar{y}^{k+1}$ into y^{k+1} so that active balance variables of y^{k+1} are ≥ 0
  set $y_p^{k+1} = \min_a ht^{y^{k+1}}_{p,a}$ for all $p \in V$ ***{imposes (3.28)}***

DUAL_FIT(y^k): $y^{fit} = \frac{y^k}{\mu f_{app}}$

Fig. 3.8: Pseudocode of the PD2$_\mu$ algorithm. The routine UPDATE_DUALS_PRIMALS is not shown because it is the same as the corresponding routine of the PD1 algorithm. The only difference is that a subset of the edges of $G_c^{x^k,\bar{y}^k}$ have different capacities (see text).

only elementary algebra to show that the primal-dual pair (x, y^{fit}) (x being the last primal solution) satisfies the relaxed slackness conditions (3.9), (3.10) with $f_1 = \mu f_{app}$, $f_2 = f_{app}$, thus leading to an f_{app}-approximate solution as well.

The main routines of PD2$_\mu$ (see Figure 3.8) are mostly similar to the ones of PD1 with only minor differences. The most important difference of the UPDATE_DUALS_PRIMALS routine lies in the assignment of capacities to those interior edges pq, qp of $G_c^{x^k,\bar{y}^k}$ whose endpoints p, q have labels $\neq c$ at the start of the current c-iteration i.e. $x_p^k = a \neq c$ and $x_q^k = b \neq c$. Then, in place of (3.19), we instead define:

$$cap_{pq} = \mu w_{pq}(d_{ac} + d_{cb} - d_{ab}) \tag{3.31}$$

$$cap_{qp} = 0 \tag{3.32}$$

Furthermore, in this case, PREEDIT_DUALS edits y^k into $\bar{y}^k$ so that:

$$\bar{y}^k_{pq,a} + \bar{y}^k_{qp,c} = \mu w_{pq} d_{ac} \tag{3.33}$$

Finally, as in PD1, the role of POSTEDIT_DUALS is again to ensure that all active balance variables are nonnegative. Note that both PREEDIT_DUALS and POSTEDIT_DUALS take care so that neither the loads nor the APF function are altered while they update the dual variables. E.g., in the case of PREEDIT_DUALS, this means that the following equalities holds true:

$$load_{pq}^{x^k,\bar{y}^k} = load_{pq}^{x^k,y^k} \tag{3.34}$$

$$APF^{x^k,\bar{y}^k} = APF^{x^k,y^k} \tag{3.35}$$

Also, note that (3.31) explains why d_{ab} must be a metric (or else it would hold $cap_{pq} < 0$).

It can then be shown that PD2$_\mu$ indeed generates an f_{app}-approximate solution as the following theorem indicates:

Theorem 3.3. *The final primal-dual solutions generated by PD2$_\mu$ are feasible and satisfy the relaxed complementary slackness conditions with $f_1 = \mu f_{app}$ and $f_2 = f_{app}$.*

Furthermore, it holds that all PD2$_\mu$ algorithms with $\mu < 1$ are non-greedy algorithms meaning that neither the primal (nor the dual) objective function necessarily decreases (increases) per iteration. Instead, it is APF which constantly decreases (see Property 1 of lemma 3.1) but since APF is always kept close to the primal function the decrease in APF is finally reflected to the values of the primal function as well. However, a notable thing happens if $\mu = 1$. In that case, due to (3.27), the load between any neighbors p, q equals exactly their separation cost (i.e. $load_{pq}^{x^k,y^k} = w_{pq} d_{x_p^k x_q^k}$) and so it can be shown that APF coincides with the primal function. In addition, PD2$_{\mu=1}$ proves to be actually equivalent to the c-expansion graph cut algorithm (that has been interpreted only as a greedy local search technique up to now). This is stated in the next theorem:

Theorem 3.4. *The label assignment x^{k+1} selected during a c-iteration of PD2$_{\mu=1}$ has the minimum primal cost among all label assignments resulting after a c-expansion*

of x^k.

Its proof is based on the following lemma in which $PRIMAL^x$ denotes the value of the primal objective function at x, while x' is any c-expansion of the current primal solution x^k:

Lemma 3.5. $PRIMAL^{x^{k+1}} = APF^{x^{k+1},\bar{y}^{k+1}} \leq APF^{x',\bar{y}^{k+1}} \leq PRIMAL^{x'}$

3.6 PD3: extending PD2 to the semimetric case

By modifying PD2$_\mu$, three different variations (PD3$_a$, PD3$_b$, PD3$_c$) may result that are applicable even if d_{ab} is a semimetric. For simplicity we will consider only the $\mu = 1$ case, i.e. only variations of PD2$_{\mu=1}$. We also recall a fact that will prove to be useful for explaining the rationale behind the algorithms' definition: (OPTIMALITY CRITERION) *the load between any* p, q *represents a virtual separation cost which should be equal to the actual separation cost of* p, q*, if the current primal-dual solutions are optimal.*

The main difficulty of extending PD2$_{\mu=1}$ to the case of a semimetric relates to all edges pq with capacity defined by (3.31) during a c-iteration, i.e. all interior edges pq whose endpoints p, q are currently assigned labels $\neq c$ (i.e. $x_p^k = a \neq c,\ x_q^k = b \neq c$), while in addition the following inequality holds: $d_{ab} > d_{ac} + d_{cb}$. Hereafter we will call any such pair (p, q) a *"conflicting pair"* and the corresponding labels (a, b, c) a *"conflicting label-triplet"*. Depending on the way we deal with such a "conflicting pair", three different variations of PD2$_{\mu=1}$ may arise.

PD3$_a$ algorithm: We choose to set $cap_{pq} = 0$ in place of (3.31). In this case it can be shown that if x^{k+1} assigns the pair of labels c, b to the objects p, q respectively then the resulting load of p, q will be $w_{pq}(d_{ab} - d_{ac})$, i.e. it will be greater than the actual separation cost $w_{pq}d_{cb}$ of p, q (this is true because (a, b, c) is a "conflicting label-triplet", thus implying that $d_{ab} > d_{ac} + d_{cb}$). Equivalently this says that the virtual separation cost of p, q overestimates their actual separation cost contrary to the OPTIMALITY CRITERION above (in all other cases one can prove that there is no such overestimation). Therefore, in this case, POSTEDIT_DUALS modifies the dual variables so that the equality between the load and the actual separation cost is restored and thus the violation of the OPTIMALITY CRITERION is

canceled by the start of the next iteration. No other differences between PD2$_{\mu=1}$ and PD3$_a$ exist.

One may also view this cost overestimation as an equivalent overestimation of the corresponding distance between labels. In the above case, for example, we saw that if labels c, b are assigned to p, q by x^{k+1}, then instead of the actual separation cost $w_{pq}d_{cb}$ the resulting overestimated cost would have been $w_{pq}\bar{d}_{cb}$ with $\bar{d}_{cb} = d_{ab} - d_{ac}$. This is equivalent to saying that the algorithm has assigned the virtual distance $\bar{d}_{cb} > d_{cb}$ to labels c, b instead of their actual distance d_{cb}. On the other hand, if (a, b) or (a, c) are assigned to p, q by x^{k+1}, then no cost overestimation takes place and so the virtual distances for these labels coincide with their actual distances i.e. $\bar{d}_{ab} = d_{ab}, \bar{d}_{ac} = d_{ac}$. *Since $\bar{d}_{ac} + \bar{d}_{cb} = \bar{d}_{ab}$ one could then argue that by replacing d with $\bar{d}$ what $PD3_a$ actually did was to overestimate the distance between labels c, b in order to restore the triangle inequality for the current "conflicting label-triplet" (a, b, c).* Put otherwise, it is as if a "dynamic approximation" of the d semimetric by a varying "metric" $\bar{d}$ is taking place with this metric $\bar{d}$ being constantly modified. Note also that for restoring the triangle inequality we could have instead designed our algorithm so that it overestimates the distance between labels a, c in place of that between c, b. Not only that, but we could have also defined an application-dependent function, say RESOLVE, which would decide (based on the current "conflicting pair") which one of the two distances (i.e. d_{ac} or d_{cb}) would be overestimated each time.

Finally, it can be shown that the primal-dual solutions generated by both PD3$_a$ and PD2$_{\mu=1}$ satisfy exactly the same conditions (3.27)-(3.30) and so PD3$_a$ is always guaranteed to lead to an f_{app}-approximate solution as well. *We can therefore conclude that PD3$_a$ directly generalizes $PD2_{\mu=1}$ to the case of a semimetric distance d_{ab}.*

PD3$_b$ algorithm: We choose to set $cap_{pq} = +\infty$ and no further differences between PD3$_b$ and PD2$_{\mu=1}$ exist. This has the following important effect: *the solution x^{k+1} produced at the current iteration can never assign the pair of labels c, b to the objects p, q respectively (due to this fact we will call labels c, b the "excluded labels"[5]).* To prove this, it suffices to recall the "reassign rule" and also observe

[5] *Note that, as in PD3$_a$, we can modify PD3$_b$ so that a function* RESOLVE *chooses which labels (i.e. (a, c) or (c, b)) are "excluded" each time. Moreover,* RESOLVE *could perhaps be defined based on a priori knowledge about each specific problem.*

that the directed edge pq can never become saturated by increasing its flow (since $cap_{pq} = +\infty$). Therefore, if label c is assigned to p by x^{k+1} (which, by the "reassign rule", means that there is an unsaturated path $s \rightsquigarrow p$) then label b can never be assigned to q, since in that case the path $s \rightsquigarrow p \rightarrow q$ would also be unsaturated (since $cap_{pq} = +\infty$) and, by the "reassign rule" again, q would have to be assigned label c as well. Put otherwise, it is as if an infinite overestimation of the distance d_{cb} between labels c, b takes place by the algorithm and so those labels are implicitly prevented from being assigned to the "conflicting pair". The price for that is that no guarantees about the algorithm's optimality can be provided. The reason is that the balance variables may now increase without bound (since $cap_{pq} = +\infty$) and so we cannot make sure that the generated dual solutions satisfy a "not too far away from feasibility" condition like (3.30). This in turn implies that no dual-fitting technique can be applied in this case. However, PD3$_b$ has a nice interpretation in the primal domain due to the following theorem:

Theorem 3.6. *The solution x^{k+1} selected by PD3$_b$ during a c-iteration has the minimum primal cost among all solutions that result after a c-expansion of x^k, except for those that assign "excluded labels" to "conflicting pairs".*

The above theorem generalizes theorem 3.4 and can be proved using similar reasoning with that theorem's proof. Furthermore, it designates the price we pay for d_{ab} being a semimetric: in the metric case we can choose the best assignment among all c-expansion moves while in the semimetric case we are only able to choose the best one among a certain subset of these c-expansion moves. Despite this fact, the considered subset contains an exponential number of c-expansion moves, which makes the algorithm a perfect candidate as a local minimizer.

Algorithm PD3$_c$: PD3$_c$ first adjusts (if needed) the dual solution y^k so that for any 2 neighbors p, q it holds: $load_{pq}^{x^k, y^k} \leq w_{pq}(d_{ac} + d_{cb})$. After this initial adjustment, which is always easy to achieve, PD3$_c$ proceeds in exactly the same way as PD2$_{\mu=1}$ except for the fact that distance d_{ab} in (3.31) is replaced with distance $\bar{d}_{ab}$ that is defined as: $\bar{d}_{ab} = \frac{load_{pq}^{x^k, y^k}}{w_{pq}}$. Obviously $\bar{d}_{ab} \leq d_{ac} + d_{cb}$ and so cap_{pq} in (3.31) is valid, i.e. $cap_{pq} \geq 0$. PD3$_c$, PD2$_{\mu=1}$ have no other differences.

It is now interesting to examine what happens if p, q is a "conflicting pair" with current labels a, b (i.e. $x_p^k = a \neq c, x_q^k = b \neq c$). In that case it also holds that

$d_{ac} + d_{cb} < d_{ab}$ and so:

$$\bar{d}_{ab} = \frac{load_{pq}^{x^k,y^k}}{w_{pq}} \leq \frac{w_{pq}(d_{ac} + d_{cb})}{w_{pq}} < \frac{w_{pq}d_{ab}}{w_{pq}} = d_{ab}$$

Furthermore, it is easy to show that if none of p, q is assigned a new label by x^{k+1} (i.e. they both retain their current labels a, b), then the resulting load will be equal to $w_{pq}\bar{d}_{ab}$, i.e. it will underestimate the actual separation cost $w_{pq}d_{ab}$ since $\bar{d}_{ab} < d_{ab}$ as was shown above (in all other cases the load will coincide with the actual separation cost).

Based on these observations, one can then see that the PD3$_c$ algorithm works in a complementary way to the PD3$_a$ algorithm: in order to restore the triangle inequality for the "conflicting label-triplet" (a, b, c), instead of overestimating the distance between either labels (c, b) or (a, c) (like PD3$_a$ did), it chooses to underestimate the distance between labels (a, b). Again one may view this as a "dynamic approximation" of the d semimetric by a constantly varying "metric" $\bar{d}$, however this time we set $\bar{d}_{ab} = \frac{load_{pq}^{x^k,y^k}}{w_{pq}} < d_{ab}$, $\bar{d}_{ac} = d_{ac}$ and $\bar{d}_{cb} = d_{cb}$.

It can be shown that the intermediate primal-dual solutions generated by algorithms PD3$_c$ and PD2$_{\mu=1}$ satisfy exactly the same conditions except for condition (3.27). In place of that condition, the intermediate solutions of PD3$_c$ satisfy:

$$load_{pq}^{x^k,y^k} \geq w_{pq}\hat{d}_{x_p^k x_q^k}, \tag{3.36}$$

where $\hat{d}_{ab} = \min_{c \in L}\big(d_{ac} + d_{cb}\big)$. By applying then the same (as in PD2$_{\mu=1}$) dual fitting factor to the last dual solution of PD3$_c$, one can easily prove that PD3$_c$ leads to an f'_{app}-approximate solution where:

$$f'_{app} = f_{app} \cdot c_0 \quad \text{with} \quad c_0 = \max_{a \neq b} \frac{d_{ab}}{\hat{d}_{ab}}. \tag{3.37}$$

Finally, we should note that if d_{ab} is a metric then PD3$_a$, PD3$_b$, PD3$_c$ all coincide with PD2$_{\mu=1}$.

3.7 Experimental results

We begin this section by first describing some unique properties of the proposed algorithms that also prove to be very useful in practice (Section 3.7.1). We then proceed and demonstrate our algorithms' effectiveness in MRF energy minimization. To this end, we apply them to a variety of low level vision tasks. In particular, here we present experimental results on the problems of stereo matching (Sections 3.7.1, 3.7.2), image restoration (Section 3.7.3), image completion (Section 3.7.3), as well as optical flow estimation (Section 3.7.4). As we show all these tasks can be formulated as particular instances of the Metric Labeling problem. Finally, to further analyze the performance of the proposed algorithms we apply them on synthetic problems and show related results in Section 3.7.5.

3.7.1 Per-instance suboptimality bounds

An important advantage of any primal-dual algorithm is that after its execution it can always tell (for free) how well it performed with respect to any given instance of Metric Labeling. In particular, as implied by the Primal-Dual Principle of section 3.3, given any pair (x, y) of integral-primal, dual-feasible solutions then the ratio $r = c^T x / b^T y$ of their costs automatically provides a new suboptimality bound in the sense that x is then guaranteed to be an r-approximation to the optimal integral solution. The minimum of these ratios constitutes the per-instance suboptimality computed by our algorithm. This leads to the following consequence that proves to be very useful in practice:
By considering the sequence of primal-dual solutions $\{x^k, y^k\}_{k=1}^t$ *generated throughout the primal-dual schema, a series of suboptimality bounds* $\{r^k = c^T x^k / b^T y^k\}_{k=1}^t$ *can be obtained. In practice, the minimum of these per-instance bounds turns out to be much tighter (i.e. much closer to 1) than the worst-case bound predicted in theory and so this allows one to have a much clearer view about the goodness of the generated solution.*

This has been verified experimentally by applying our algorithms to the stereo matching problem. In this case, labels correspond to image pixel disparities and they can be chosen from a set $L = \{0, 1, \ldots, K\}$ of discretized disparities where K denotes the maximum allowed disparity. The vertices of the graph G are the

image pixels and the edges of G connect each pixel to its 4 immediate neighbors in the image. During our tests, the label cost for assigning disparity a to the image pixel p has been set equal to:

$$c_{p,a} = |I_{right}(p-a) - I_{left}(p)| \tag{3.38}$$

where I_{left}, I_{right} represent the intensities of the left and right images respectively.

We have applied our algorithms to the well-known Tsukuba stereo data set [115] setting the maximum disparity value equal to $K = 14$, based on the provided ground truth data. A sample from the results produced when using our algorithms are shown in Fig. 3.9. It should be noted that no attempt has been made to model occlusions during the stereo matching procedure, while, in addition, all edge weights w_{pq} have been set equal to each other instead of properly adjusting their values based on image intensity edges (which would improve the results considerably for this specific example). The reason for that as well as for using the very simple label cost presented in (3.38) is because our main goal was not to produce the best possible disparity estimation, but only to test the tightness of the suboptimality bounds that are provided by the considered algorithms, i.e. to test the effectiveness of these algorithms in minimizing the objective function.

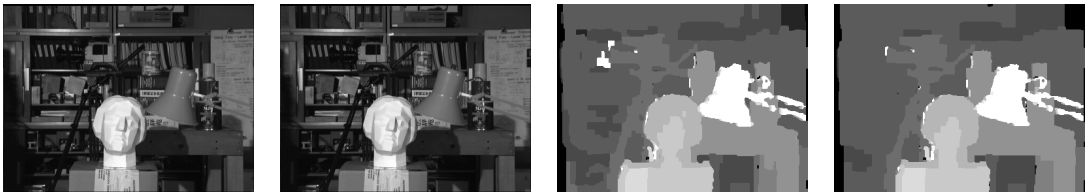
To this end, 3 different distances d_{ab} have been used during our experiments. These are the Potts distance $d_{1,ab}$ (a metric), the truncated linear distance $d^{\lambda}_{2,ab}$ (also a metric) and the truncated quadratic distance $d^{\lambda}_{3,ab}$ (a semimetric), defined as follows:

$$d_{1,ab} = 1 \qquad \forall a \neq b \tag{3.39}$$

$$d^{\lambda}_{2,ab} = \min(\lambda, |a-b|) \qquad \forall a, b \tag{3.40}$$

$$d^{\lambda}_{3,ab} = \min(\lambda, |a-b|^2) \qquad \forall a, b \tag{3.41}$$

In the above equations the constant λ denotes the maximum allowed distance. Each experiment consisted of selecting an approximation algorithm and a distance function and then using them for computing disparities for each one of the Tsukuba stereo pairs. The average values of the obtained suboptimality bounds are displayed in table 3.1. The columns f^{PD1}_{app}, $f^{PD2_{\mu=1}}_{app}$, $f^{PD3_a}_{app}$, $f^{PD3_b}_{app}$, $f^{PD3_c}_{app}$ of that table list these averages for the algorithms $PD1, PD2_{\mu=1}, PD3_a, PD3_b$ and



(a) (b) (c) (d)

Figure 3.9: (a) The left and **(b)** right images for one stereo pair from the Tsukuba data set. **(c)** The disparity estimated by the PD1 algorithm. **(d)** and the PD2$_{\mu=1}$ algorithm. The Potts distance (a metric) has been used in this example and so PD3$_a$, PD3$_b$, PD3$_c$ produce the same result with PD2$_{\mu=1}$.

$PD3_c$ respectively. In addition, the last column lists the value of the corresponding approximation factor f_{app} which, as already proved, makes up a worst-case suboptimality bound for most of the above algorithms. By observing table 3.1 one can conclude that the per-instance suboptimality bounds are much tighter (i.e. much closer to 1) than the worst-case bounds predicted in theory. This holds for all cases, i.e. for all combinations of algorithms and distances, and thus indicates that *the presented algorithms are always able to extract a nearly optimal solution (with this being true even for the more difficult case where d_{ab} is merely a semimetric).*

Besides the tightness of the per instance suboptimality bounds, another important issue is their accuracy, i.e. how well these bounds predict the true suboptimality of the generated solutions. To investigate this issue we modified our experiments in the following way: we applied our stereo matching algorithms to one image scanline at a time (instead of the whole image). In this case the graph G reduces to a chain and the true optimum can be easily computed using dynamic programming. This in turn implies that we are able to compute the true suboptimality of a solution. By using this fact we have thus constructed table 3.2. Its columns f_{true}^{PD1}, $f_{true}^{PD2_{\mu=1}}$, $f_{true}^{PD3_a}$, $f_{true}^{PD3_b}$, $f_{true}^{PD3_c}$ contain the true average suboptimality of the solutions of $PD1, PD2_{\mu=1}, PD3_a, PD3_b$ and $PD3_c$ respectively, where the average is taken over all image scanlines. By examining that table 3.2 one may easily conclude that the true suboptimality of a solution is always close to the corresponding estimated suboptimality bound. This means that these bounds are relatively accurate and therefore reliable for judging the goodness of an algorithms's solution. Furthermore, in this way, we can always decide if a bad

Distance	$\mathbf{f_{app}^{PD1}}$	$\mathbf{f_{app}^{PD2_{\mu=1}}}$	$\mathbf{f_{app}^{PD3_a}}$	$\mathbf{f_{app}^{PD3_b}}$	$\mathbf{f_{app}^{PD3_c}}$	$\mathbf{f_{app}}$
Potts	1.0104	1.0058	1.0058	1.0058	1.0058	2
Trunc. Linear $^{\lambda=5}$	1.0226	1.0104	1.0104	1.0104	1.0104	10
Trunc. quad. $^{\lambda=5}$	1.0280	-	1.0143	1.0158	1.0183	10

Table 3.1: Average suboptimality bounds (colums 2-6) obtained for the Tsukuba data set. As expected they are much closer to 1 than the theoretical suboptimality bounds f_{app} listed in the last column and thus a nearly optimal solution is obtained in all cases. Note that $PD2_{\mu=1}$ can be applied only if d_{ab} is a metric and in that case PD2$_{\mu=1}$, PD3$_a$, PD3$_b$ and PD3$_c$ (as well as their bounds) coincide.

generated solution is the result of a bad optimization procedure or a bad modeling of the problem at hand.

For the Tsukuba sequence, on average 4 outer iterations (or equivalently $60 = 4 \cdot 15$ inner iterations) are needed for the algorithms to terminate. The corresponding running time is 46 secs (measured on a 2.4GHz CPU). The plots in Figure 3.10 show how the primal-dual ratios vary during the execution of our algorithms (for the Tsukuba data set). For the first two plots a metric distance between labels has been used, while for the last one a semimetric distance has been chosen. It is worth noticing how rapidly the primal-dual ratios drop in all cases. They come very close to 1 just after a few inner iterations, meaning that the algorithms converge really fast, while computing an almost optimal solution at the same time. Based on this observation one may also use the values of these ratios to control the algorithms' convergence (e.g. if the ratios are close to 1 and do not vary too much per iteration one may decide that convergence has been

Distance	$\mathbf{f_{app}^{PD1}}$	$\mathbf{f_{true}^{PD1}}$	$\mathbf{f_{app}^{PD2_{\mu=1}}}$	$\mathbf{f_{true}^{PD2_{\mu=1}}}$	$\mathbf{f_{app}^{PD3_a}}$	$\mathbf{f_{true}^{PD3_a}}$	$\mathbf{f_{app}^{PD3_b}}$	$\mathbf{f_{true}^{PD3_b}}$	$\mathbf{f_{app}^{PD3_c}}$	$\mathbf{f_{true}^{PD3_c}}$
Potts	1.0098	1.0036	1.0066	1.0004	1.0066	1.0004	1.0066	1.0004	1.0066	1.0004
Trunc. Linear	1.0202	1.0107	1.0115	1.0021	1.0115	1.0021	1.0115	1.0021	1.0115	1.0021
Trunc. quad.	1.0255	1.0130	-	-	1.0135	1.0011	1.0144	1.0020	1.0160	1.0036

Table 3.2: The average suboptimality bounds (columns 2-4-6-8-10) obtained when applying our stereo matching algorithms to one scanline at a time (instead of the whole image). In this case, we are also able to compute the true average suboptimality (columns 3-5-7-9-11) of the generated solutions using dynamic programming. As can be seen by inspecting the table the suboptimality bounds always approximate the true suboptimality relatively well, meaning that they can be safely used as a measure for judging the goodness of a generated solution.

reached). This way one may further reduce the running time of the algorithm.

3.7.2 Stereo matching

Besides the Tsukuba dataset we have also applied our algorithms to image pairs from the SRI tree image sequence (Fig. 3.11(a)). The selected pairs had a maximum disparity of 11 pixels. Given our algorithms' ability to handle both metric and semimetric distances equally well, the following semimetric has been used in this case: $d^{\kappa,\lambda}_{4,ab} = |a-b|$ if $|a-b| <= \kappa$, otherwise $d^{\kappa,\lambda}_{4,ab} = \lambda$. We always assume $\kappa < \lambda$. In this specific example we have used $(\kappa, \lambda) = (2, 10)$. The rationale behind this distance is that it assigns a low penalty to small (i.e. $\leq \kappa$) changes in disparity (thus allowing surfaces with smoothly varying disparity like the slanted ground in the SRI image), but assigns a high penalty λ to large disparity gaps. Despite the fact that $d^{\kappa,\lambda}_{4,ab}$ is a semimetric our algorithms did not face any problem in efficiently minimizing the corresponding objective function and thus localizing the trees as well as the slanted ground in the SRI image. The resulting disparity is shown in Figure 3.11(b). The average running time to convergence has been 33 secs. We have also applied the α-β-swap algorithm [27] to the SRI dataset using exactly the same settings. Although this graph-cut based algorithm is applicable even in the case of a semimetric label distance, its disadvantage is that it may get trapped to a bad local minimum, i.e. it cannot make any guarantees about the optimality of the solutions it generates. This is indeed the case here since, despite the fact that exactly the same objective function has been minimized by both algorithms, the final energy produced by α-β-swap was 8.3% higher than the energy estimated by our method. The corresponding disparity is shown in Figure 3.11(c).

As a further example we illustrate how one could favor disparities that are not violating the uniqueness constraint, just by use of an appropriate non-metric distance d_{ab}. This can possibly lead to a better handling of occlusions as well in some cases. To this end, an extra label for occlusions, say $\hat{o}$, is introduced first whose label cost is equal to $c_{\hat{o}}$ for all pixels i.e. $c_{p,\hat{o}} = c_{\hat{o}}$. Assuming without loss of generality that image scanlines coincide with the epipolar lines we then introduce additional horizontal edges in the graph G: we connect any pixel (x, y) in the left image to the K pixels to its right $(x+1, y)$, ...,$(x+K, y)$, where K is the

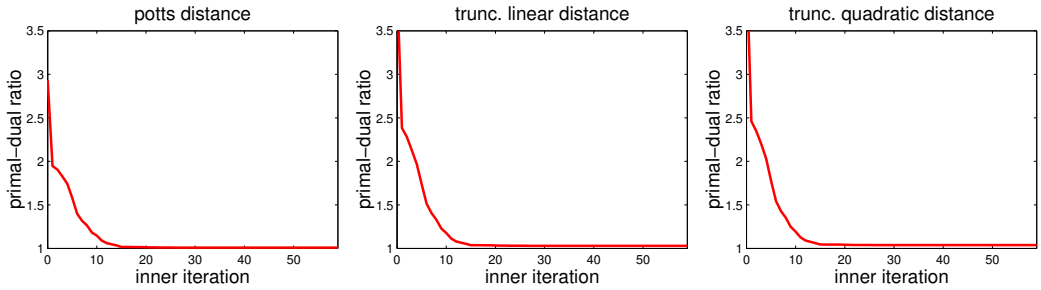
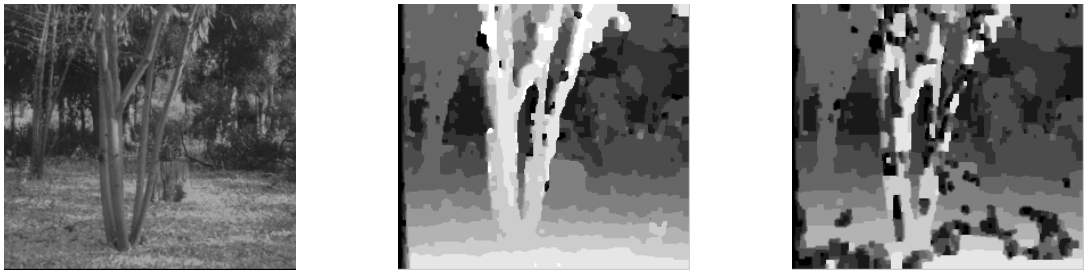


Figure 3.10: These 3 plots show how the primal-dual ratios vary during the first 4 outer iterations (or equivalently the first $60 = 4{\cdot}15$ inner iterations) using the Tsukuba sequence as input. **(Left)** The potts metric, **(Middle)** the trunc.linear metric and **(Right)** the trunc. quad. semimetric have been used respectively as label distance d_{ab}. Notice how rapidly the ratios drop in all cases (i.e. they get very close to 1 just after a few inner iterations).

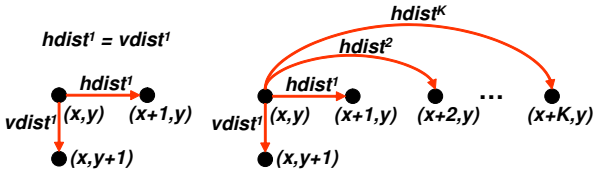


(a) (b) (c)

Fig. 3.11: (a) One image from the SRI tree image sequence. **(b)** Computed disparities when using PD3$_a$ and the semimetric $d^{\kappa,\lambda}_{4,ab}$ with $(\kappa, \lambda) = (2, 10)$. **(c)** Disparities computed by the α-β-swap algorithm using the same semimetric. The solution of α-β-swap has 8.3% higher energy than the corresponding solution of the $PD3_a$ algorithm despite the fact that both algorithms try to minimize exactly the same energy function.

maximum disparity (see Fig. 3.12(b)). For measuring the separation cost between the labels of (x, y), $(x + k, y)$ we will use the distance function $hdist^k$. We will therefore use K different distance functions in total for all the horizontal edges. On the other hand, no additional vertical edges are introduced and so any pixel will be connected only to its immediate vertical neighbors as before with $vdist^1$ denoting the common distance function for all these edges.

Distances $hdist^1$, $vdist^1$ (which are related to edges connecting pixels adjacent in the image) will be used for enforcing the smoothness of the disparity field as before. E.g. both can be set equal to the potts metric: $hdist^1_{ab} = vdist^1_{ab} = d_{1,ab}$. The rest of $hdist^k$ will be used just for assigning an extra penalty M to all pairs of labels violating the uniqueness constraint. For all other pairs of distinct labels,



(a) No uniqueness constraint

(b) Edges of G in the case of favoring the uniqueness constraint

Percentage of pixels with absolute error > 1 in non-occluded regions	1.14%
Percentage of missed occluded pixels in occluded regions	12.31%

(c) Error statistics for the *'map'* pair

Figure 3.12: (a) Uniqueness constraint is not favored and the graph G coincides with the image grid. A common label distance is used for all edges (i.e. $hdist^1 = vdist^1$) **(b)** To favor the uniqueness constraint we introduce additional horizontal edges in G which connect any pixel with the K pixels to its right (K is the maximum disparity).

$hdist^k$ then simply assigns a very small distance ε (with $\varepsilon \ll M$):

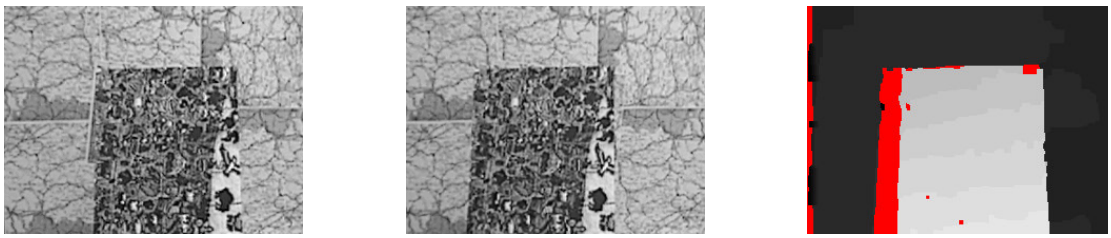
$$b = a + k \Rightarrow hdist^k_{ab} = M, \quad b \neq a + k \;\&\; b \neq a \Rightarrow hdist^k_{ab} = \varepsilon, \quad b = a \Rightarrow hdist^k_{ab} = 0.$$

A result of applying this distance (with $c_{\delta} = 23, M = 10, \varepsilon = 0.01$) to the *map* stereo pair appears in Fig. 3.13. Error statistics are displayed in Fig. 3.12(c).

3.7.3 Image restoration and image completion

In image restoration, we are given as input a corrupted (by noise) image and the objective is to extract the original (uncorrupted) image. In this case the labels represent intensities (or colors) while the label cost for assigning intensity a to pixel p can be set equal to: $c_{p,a} = |I(p) - a|$, where I represents the array of intensities of the input image. The graph G that will be used when solving the Metric Labeling problem coincides again with the image grid.

The example of Fig. 3.14 illustrates the importance of using semimetric distances d_{ab} on the task of image restoration as well. The original image (Fig. 3.14(a))

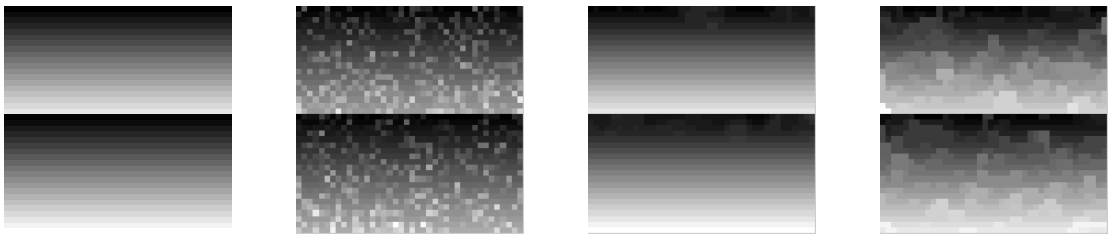


(a) The left *'map'* image

(b) The right *'map'* image

(c) Estimated disparity

Fig. 3.13: Red pixels indicate occlusions



(a) (b) (c) (d)

Fig. 3.14: (a) Original uncorrupted image. **(b)** Noisy input image. **(c)** Restored image using semimetric $d^{\kappa,\lambda}_{4,ab}, (\kappa,\lambda) = (2,30)$ **(d)** Restored image using the truncated linear metric $d^{\lambda}_{2,ab}$, λ=30.

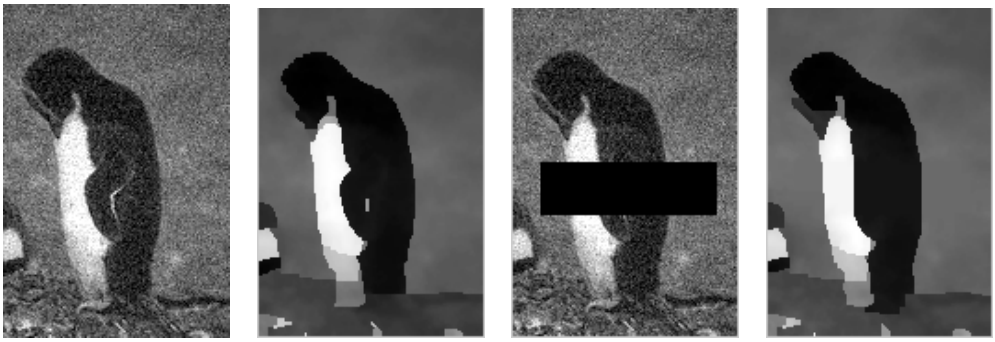
Label distance d_{ab}	**pixels with intensity errors**	**average intensity error**
semimetric	8.2%	0.21
metric	41.4%	1.12

Table 3.3: Error statistics for the image restoration example of Fig. 3.14

consists of 2 identical patterns placed vertically. Each pattern's intensity is kept constant along the horizontal direction and increases linearly with step 2 from top to bottom. The input image is then formed by corrupting the original image with white noise (Fig. 3.14(b)). Although our algorithms managed to restore the original image with only a few errors by use of the $d^{\kappa,\lambda}_{4,ab}$ semimetric (Fig. 3.14(c)), this was not the case when the truncated linear metric $d^{\lambda}_{2,ab}$ (or the potts metric) has been used, despite the tweaking of the λ parameter. The best obtained result with such a metric (after tweaking λ) is shown in Fig. 3.14(d). The error statistics for this restoration example are shown in table 3.3.

Another semimetric which is very commonly used in image restoration problems is the truncated quadratic distance $d^{\lambda}_{3,ab} = min(|a-b|^2, \lambda)$. That distance with $\lambda = 200$ was used in the restoration of the contaminated (with Gaussian noise) image of Fig. 3.15(a). In this case the following function (which is more robust against outliers) has been used for the label costs: $c_{p,a} = \lambda_0 min(|I(p) - a|^2, \lambda_1)$ with $\lambda_0 = 0.05, \lambda_1 = 10^4$. Notice that our algorithm managed not only to remove the noise completely (see Fig. 3.15(b)), but also to maintain the boundaries of the objects at the same time.

The same semimetric (i.e. the truncated quadratic distance) can be also used for the task of image completion. Besides containing Gaussian noise, the image in Fig. 3.15(c) also has a part which has been masked. The labels costs of masked



(a) Noisy input image (b) Restored image (c) Noisy input image with masked pixels (d) Restored and completed image

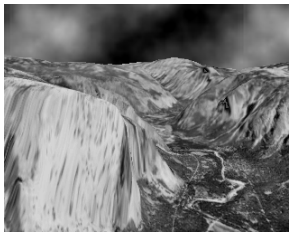
Fig. 3.15: Examples of image restoration and image completion

pixels have been set to zero, while for the rest of the pixels the costs have been set as before. As can be seen from Fig. 3.15(d), our algorithm managed not only to remove the noise again, but also to fill the missing part in a plausible way.

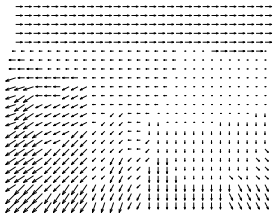
3.7.4 Optical flow estimation

Global methods [19,28] estimate optical flow u_x, u_y by minimizing a functional of the form: $E(u_x, u_y) = \int_I \rho_D(I_x u_x + I_y u_y + I_t) + \lambda \cdot \rho_S(\sqrt{|\nabla u_x|^2 + |\nabla u_y|^2})dxdy$ where I_x, I_y, I_t denote spatial and temporal image derivatives while ρ_D, ρ_S denote penalty functions. By discretizing $E(u_x, u_y)$ we can easily incorporate all such methods into our framework: the 1^{st} term (which expresses the *optic flow constraint equation*) and the 2^{nd} term (which is a regularizer) will then correspond to the label costs and separation costs respectively. Furthermore, due to our weak assumptions on d_{ab}, our framework allows us to set ρ_S equal to any of the so-called *robust penalty functions* [19] (e.g. the Lorentzian $\rho_S(x) = \log(1 + \frac{1}{2}(x/\sigma)^2)$), which are known to better cope with outliers or flow discontinuities. Due to this fact our framework can also incorporate the state-of-the-art combined local-global method (CLG) [28], which just replaces I_x, I_y, I_t (in the 1^{st} term of the above functional) with a structure tensor. This is important since our algorithms can always compute a solution near the global minimum and so by using them as initializers to CLG (or to any other global method) we can help such methods to avoid a local minimum.

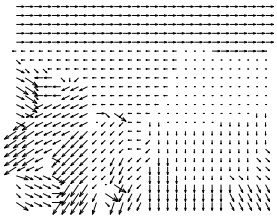
Besides using the *optic flow constraint equation* in our label costs, our frame-



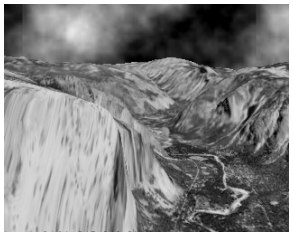
(a) 4^{th} frame of *yosemite* sequence with clouds



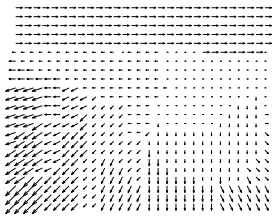
(b) Our flow: 6.97^o avg. angular error, 4 iterations



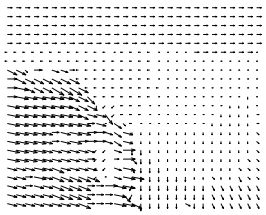
(c) α-β-swap flow: 14.73^o avg. angular error, 11 iterations, 19.2% higher energy than (b)



(d) 11^{th} frame of *yosemite* sequence with clouds



(e) Our flow: 6.91^o avg. angular error, 4 iterations



(f) α-β-swap flow: 39.29^o avg. angular error, 23 iterations, 56.7% higher energy than (e)

Fig. 3.16: Estimated flow between frames 4, 5 (1^{st} row) and 11, 12 (2^{nd} row) of *yosemite* sequence. Although more outer iterations were used by α-β-swap, its optical flow had 19.2% and 56.7% higher energy than our optical flow.

work also allows the use of other label costs. E.g. we can set $c_{p,a} = |I_1(p+a) - I_0(a)|$ where I_0, I_1 are the 1^{st} and 2^{nd} image. In this case, due to the two-dimensional nature of optical flow, it is important that not only the magnitudes, but especially the directions of the optical flow vectors are estimated correctly as well. To this end the following semimetric distance between labels can be used:

$$d_{ab} = \text{dist}_{ab} + \tau \cdot \text{angledist}_{ab}$$

Here dist_{ab} denotes a truncated euclidean distance between the optical flow vectors a, b, i.e. $\text{dist}_{ab} = min(||a - b||, M)$, while the 2^{nd} term is used for giving even more weight to the correct estimation of the vectors' direction. In particular, it penalizes (in a robust way) abrupt changes in the direction of the vectors a, b and is defined as follows:

$$\text{angledist}_{ab} = \begin{cases} 1, & \text{if } \text{angle}_{ab} > 45^o \\ 0, & \text{otherwise} \end{cases}$$

where angle_{ab} denotes the angle (in degrees) between vectors a and b. We have applied both our algorithm and the α-β-swap algorithm to the well known *yosemite*

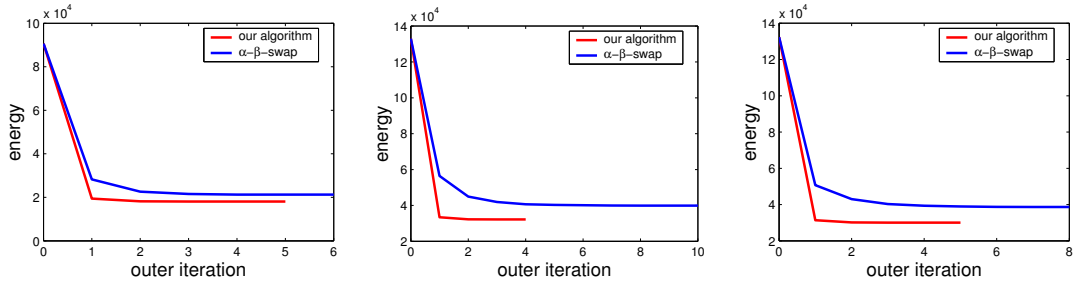
image sequence using as label distance the above semimetric with parameters $M = 5, \tau = 5$. The results as well as error statistics are shown in Figure 3.16. *What is important to note here is that although both algorithms are trying to minimize exactly the same objective function, the resulting solutions of α-β-swap have much higher energy.* It seems that, contrary to our method, α-β-swap is not powerful enough to escape from bad local minima in this case.

3.7.5 Synthetic problems

To further examine the ability of our algorithms to optimize the energy of an MRF, we also tested them on a set of synthetic problems. In these problems the vertices of a 30×30 grid were chosen as the nodes of the graph G while the total number of labels was set equal to K. The label costs for all nodes were generated randomly by drawing samples from a uniform distribution in the $[\varrho_0\ \varrho_1]$ interval, while for measuring the distance between labels a random semimetric has been used that was constructed as follows: equal labels were assigned zero distance whereas the distance for different labels was generated randomly in the $[\varrho_0\ \varrho_1]$ interval again.

Three experiments have been conducted: in the 1^{st} one, a random spanning tree of the 30×30 grid was used as the graph G and the number of labels was $K = 60$ while in the 2^{nd} and 3^{rd} experiment the graph G had inherited the structure of the underlying grid and the number of labels was $K = 60$ and $K = 180$ respectively. For each experiment 100 random problems were constructed (all with $\varrho_0 = 1, \varrho_1 = 100$) and the resulting average energies per outer iteration, for both our algorithm and the α-β-swap algorithm, are shown in the plots of Figure 3.17. Notice that, compared to α-β-swap, our algorithm manages to produce a solution of lower energy in all cases. At the same time it needs less iterations to converge. This behavior is a typical one and has been observed in real problems as well. Notice also that as the number of labels or the graph complexity increases, the gap in performance between the 2 algorithms increases as well.

The efficiency of our algorithms in the case where d_{ab} is a semimetric can be also illustrated by the synthetic example of Figure 3.18. Although $PD3_a$, $PD3_b$ and $PD3_c$ are always able to locate the exact global minimum for this example, the a-b swap algorithm may get stuck at a local minimum that can be arbitrarily



(a) 1^{st} experiment: graph G is a tree, $K = 60$ labels (b) 2^{nd} experiment: graph G is a grid, $K = 60$ labels (c) 3^{rd} experiment: graph G is a grid, $K = 180$ labels

Fig. 3.17: α-β-swap produces an energy which is higher by **(a)** 17%, **(b)** 23% and **(c)** 28% with respect to our algorithm's energy. Notice that as the number of labels increases the gap in performance increases as well.

far from the true minimum.

Label costs

$C_{p,a}$	*p*	*q*	*r*
a	*0*	*T*	*T*
b	*T*	*0*	*T*
c	*2*	*2*	*0*

Label distance

d_{ab}	*a*	*b*	*c*
a	*0*	*T/2*	*T*
b	*T/2*	*0*	*T/2*
c	*T*	*T/2*	*0*

Labeling A (Local minimum)

p	*q*	*r*
a	*b*	*c*

Labeling B (Global minimum)

p	*q*	*r*
c	*c*	*c*

Fig. 3.18: A synthetic example where the graph G has 3 vertices $\{p, q, r\}$ and 2 edges $\{pq, qr\}$ while the labels L are $\{a, b, c\}$. Label costs c_{pa} and the distance d_{ab} (a semimetric) are shown. The α-β-swap algorithm can get stuck in labeling A whose cost is T i.e. arbitrarily larger than the true minimum cost which is 4 (labeling B). On the contrary $PD3_a$, $PD3_b$, $PD3_c$ can always locate the optimal labeling B. Example taken from [27].

3.8 Conclusions

A new theoretical framework has been proposed for both understanding and developing algorithms for the approximate optimization of MRFs with both metric and non-metric energy functions. This set of MRFs can capture a very important class of problems in vision. The above framework includes the state-of-the-art α-expansion algorithm merely as a special case (for metric energy functions). Moreover, it provides algorithms which have guaranteed optimality properties even for the case of semimetric potentials. In fact, in all cases our primal-dual algorithms are capable of providing per-instance suboptimality bounds which, in practice, prove to be very tight (i.e. very close to 1) meaning that the resulting solutions are nearly optimal. The theoretical setting of the proposed framework rests on duality

theory of linear programming, which is entirely different than the setting of the original graph-cut work. This way an alternative and more general view of the very successful graph-cut algorithms for approximately optimizing MRFs is provided which is an important advance. We strongly believe that this more general view of graph cut techniques may give rise to new related research, which could lead to even more powerful MRF optimization algorithms in the future. Moreover, a novel optimization technique, the primal-dual schema, has been introduced to the field of computer vision and the resulting algorithms have proved to give excellent experimental results on a variety of low level vision tasks, such as stereo matching, image restoration, image completion and optical flow estimation. With respect to metric MRFs, the PD2$_{\mu=1}$ has given very good results experimentally. On the other hand, for the case of semimetric MRFs, the algorithms PD3$_a$, PD3$_c$ gave the best results in practice. Therefore, based on the fact that any of the PD3 algorithms reduces to the algorithm PD2, if the distance function is a metric, one may choose to implement either one of the PD3$_a$, PD3$_c$ algorithms. Finally, we should note that for certain special cases of the ML problem, our algorithms' theoretical approximation factors coincide with the so-called *integrality gap* of the linear program in (3.1), which is essentially the best possible approximation factor a primal-dual algorithm may achieve [136]. E.g. such is the case with the Generalized Potts model, whose integrality gap is known to be 2 [75], i.e. equal to f_{app}. This explains in yet another way why graph-cut techniques are so good in optimizing problems related to the Potts energy. In conclusion, a new powerful optimization tool has been added to the arsenal of computer vision, capable of tackling a very wide class of problems.

CHAPTER 4

Priority-BP and the Problem of Image Completion

The main goal of this chapter is to present a new MRF optimization method, called Priority-BP, which significantly extends standard Belief Propagation. For reasons of concreteness, we prefer not to examine Priority-BP in an abstract way but, instead, we choose to directly test the algorithm by applying it to the very difficult problem of image completion, that has received growing attention in recent years. Based on this observation, the contributions that are made in this chapter are twofold:

On one hand, as already mentioned above, a novel optimization scheme (Priority-BP) is proposed. Unlike previous work, it carries 2 important extensions over standard BP: priority-based message scheduling and dynamic label pruning. Together, these two extensions manage to resolve what is currently considered as the major limitation of Belief Propagation: its inefficiency in handling MRFs with very large discrete state-spaces. Moreover, both extensions are generic and do not make use of any domain-specific knowledge. They can therefore be applied to any MRF, i.e. a very wide class of problems in computer vision. It is thus the first time, at least to the best of our knowledge, that a framework which is targeted for general MRFs manages to resolve the above mentioned major limitation of Belief Propagation.

On the other hand, to show the effectiveness of Priority-BP, a novel exemplar-based framework is proposed, which treats the problems of image completion, texture synthesis and image inpainting in a unified manner. Contrary to most of the existing methods, all of these tasks are posed as discrete MRF optimization problems with a well-defined global objective function. Furthermore, visually inconsistent results due to greedy patch assignments are avoided, as our method manages to maintain throughout its execution many candidate source patches for each block of missing pixels. This is unlike the majority of current methods, which greedily fill

patches and keep them fixed thereafter. In addition, the effectiveness of our method is demonstrated on a wide variety of difficult image completion examples.

The chapter starts with a description of the image completion problem as well as a review of related work. It then continues with a presentation of our novel exemplar based framework. The Priority-BP algorithm is described next, while, finally, it is shown how that algorithm can be used in the problem of completing images.

When I'm working on a problem, I never think about beauty. I think only how to solve the problem. But when I have finished, if the solution is not beautiful, I know it is wrong.

—Richard Buckminster Fuller (1895-1983)

4.1 Introduction

Image completion is a problem that has attracted a considerable amount of attention over the last years. The goal of image completion is to fill the missing part of an incomplete image in such a way that a visually plausible outcome is obtained. Ideally, one would like to be able to apply the image completion process to:

- complex natural images
- with (possibly) large missing parts
- in a fully automatic manner

For the above reasons, this is a very challenging problem. Moreover, it can have many applications, e.g. in image editing, film post-production, image restoration etc. (see Figure 4.1).

There have been three main approaches for dealing with the image completion problem so far:

- statistical based methods
- PDE based methods
- as well as exemplar based methods

Fig. 4.1: *Object removal* is just one example of the many uses of image completion. In the specific example shown above, the user wants to remove a person from the input image on the left. He therefore simply marks a region around that person and that region must then be filled automatically so that a visually plausible outcome is obtained.

In order to briefly explain the main limitations of current state-of-the-art methods for image completion, we provide a short review of related work for each one of the three classes mentioned above.

Statistical based methods: These methods try to describe textures through the use of compact parametric statistical models. E.g. Portilla and Simoncelli [107] use joint statistics of wavelet coefficients for that purpose, while Heeger and Bergen [61] make use of color histograms at multiple resolutions for the analysis of the textures. Parametric statistical models have been also proposed for the case of image sequences. E.g. Soatto et al. [114] have proposed the so-called *dynamic texture* model, while a similar idea has been also described by Fitzgibbon in [49]. A parametric representation for image sequences had been previously presented by Szummer and Picard [130] as well. These parametric models for video have been mainly used for modeling and synthesizing dynamic stochastic processes such smoke, fire or water.

However, the main drawback of all the methods that are based on parametric statistical models is that they are applicable only to the problem of texture synthesis and not to the general problem of image completion. But even in the restricted case of texture synthesis, they can synthesize only textures which are highly stochastic and usually fail to do so for textures containing structure as well. Nevertheless, in cases where parametric models are applicable, they allow greater flexibility with respect to the modification of texture properties. E.g. Doretto and Soatto [39] can edit the speed as well as other properties of a video texture by modifying the parameters of the statistical model they are using (which is a linear dynamical system in their

case). Furthermore, these methods can be very useful for the process which is reverse to texture synthesis, i.e. the analysis of textures .

PDE based methods: These methods, on the other hand, try to fill the missing region of an image through a diffusion process by smoothly propagating information from the boundary towards the interior. According to these techniques, the diffusion process is simulated by solving a partial differential equation (PDE) which is typically non-linear and of high order. This class of methods has been first introduced by Bertalmio et al. in [13], in which case the authors try to fill a hole in an image by propagating image Laplacians in the isophote direction. Their algorithm tries to mimic the behavior of professional restorators in image restoration. In another case, the partial differential equations that have been employed for the image filling process were related to the Navier-Stokes equations in fluid dynamics [12], while Ballester et al. [7] have derived their own partial differential equations by formulating the image completion problem in a variational framework. Furthermore, recently, Bertalmio et al. [14] have proposed to decompose an image into two components. The first component is representing structure and is filled by using a PDE based method, while the second component represents texture and is filled by use of a texture synthesis method. Finally, Chan and Shen [32] have used an elastica based variational model for the process of image filling, while Bertalmio has also proposed the use of a nonlinear PDE of third order in order to achieve (what he calls) contrast invariant inpainting [11].

However, the main disadvantage of almost all PDE based methods is that they are mostly suitable for image inpainting situations. This term usually refers to the case where the missing part of the image consists of thin, elongated regions. Furthermore, PDE-based methods implicitly assume that the content of the missing resion is smooth and non-textured. For this reason, when these methods are applied to images where the missing regions are large or textured, they usually oversmooth the image and introduce blurring artifacts (see Figure 4.2). On the contrary, we would like our method to be able to handle images that contain possibly large missing parts. In addition to that, we would also like our method to be able to fill arbitrarily complex

(a) Original image (b) Image with missing region (c) Completion using image inpainting

Fig. 4.2: Image inpainting methods, when applied to large or textured missing regions, oversmooth the image and introduce blurring artifacts.

natural images, i.e. images containing texture, structure or a combination of both.

Exemplar-based methods: Finally, the last class of methods consists of the so-called *exemplar-based* techniques, which actually have been the most successful techniques up to now. These methods try to fill the unknown region simply by copying content from the observed part of the image. Starting with the seminal work of Efros and Leung in [43], these methods have been mainly used for the purpose of texture synthesis. All exemplar-based techniques for texture synthesis that have appeared until now were either pixel-based [22, 143] or patch-based [84, 89, 148], meaning that the final texture was synthesized one pixel or one patch at a time (by simply copying pixels or patches from the observed image respectively). Somewhere in between is the method of Ashikhmin [6] where a pixel-based technique, that favors the copy of coherent patches, has been used in this case. Usually, patch-based methods achieve results of higher quality since they manage to implicitly maintain higher order statistics of the input texture. Among patch-based methods, one should mention the work of Kwatra et al. [84] who manage to synthesize a variety of textures by making use of computer vision graph-cut techniques. Another interesting work is that of Hertzmann et al. [63] where the authors try to automatically learn painting styles from training data that

consist of input-output image pairs. The painting styles, once learnt, can then be applied to new input images. Also, Efros and Freeman [42] perform what they call texture transfer, i.e. rendering an object with a texture taken from a different object. Exemplar-based methods for texture synthesis have been also used for the case of video. E.g. Schodl et al. [117] are able to synthesize new video textures simply by rearranging the recorded frames of an input video while the texture synthesis method of Kwatra et al. [84], that has been mentioned above, applies to image sequences as well. Also, Patwardhan et al. [102] use a video inpainting method for filling-in missing parts of a video sequence taken from a static camera, while Bhat et al. [17] have built an interactive system for synthesizing and editing video of natural phenomena that exhibit continuous flow patterns.

As already mentioned earlier, exemplar-based methods have been mainly used for the purpose of texture synthesis up to now. Recently, however, there have been a few authors who have tried to extend these methods to image completion as well. But, in this case, a major drawback of related approaches stems from their greedy way of filling the image, which can often lead to visual inconsistencies. Some techniques try to alleviate this problem by asking assistance from the user instead. *E.g* Jian Sun *et al* [127] require the user to specify the curves on which the most salient missing structures reside (thus obtaining a segmentation of the missing region as well), while Drori *et al* [40] use what they call "points of interest". Also, some other methods [70] rely on already having a segmentation of the input image. But it is a well known fact that natural images segmentation is an extremely difficult task and, despite extensive research, no general method for reliably solving it currently exists. Some other methods [83, 146] are preferring to take a more global approach and formulate the problem in a way that a deterministic EM-like optimization scheme has to be used for image completion. It is well known, however, that expectation-maximization schemes are particularly sensitive to the initialization and may get easily trapped to poor local minima (thus violating the spirit of a global approach). For fixing this problem, one has to use multiscale image completion, but this is still not always safe. *E.g* any errors that may occur during the image completion

process at the coarse scale, will probably carry through at finer scales as well. Finally, recent exemplar-based methods also place emphasis on the order by which the image synthesis proceeds, usually using a confidence map for this purpose [36, 40]. However, two are the main handicaps of related existing techniques. First, the confidence map is computed based on heuristics and ad hoc principles that may not apply in the general case and second, once an observed patch has been assigned to a missing block of pixels, that block cannot change its assigned patch thereafter. This last fact reveals the greediness of these techniques, which may again lead to visual inconsistencies.

In order to overcome all the limitations of the above mentioned methods a new exemplar-based approach for image completion is proposed, which makes the following contributions:

1. Contrary to greedy synthesis methods we pose image completion as a discrete global optimization problem with a well defined objective function.

2. Our formulation applies not only to image completion, but also to texture synthesis and image inpainting thus providing a unified framework for all of these tasks.

3. No user intervention is required by our method which manages to avoid greedy patch assignments by maintaining (throughout its execution) many candidate source patches for each block of missing pixels.

4. To this end a novel optimization scheme is proposed, the "Priority-BP" algorithm, which carries 2 major improvements over standard belief propagation: *"dynamic label pruning"* and *"priority-based message scheduling"*. Together they bring a dramatic reduction in the overall computational cost of BP, which would otherwise be intolerable due to the huge number of existing labels. We should finally note that both extensions are generic and can be used for the optimization of any MRF (*i.e* a wide class of problems in vision).

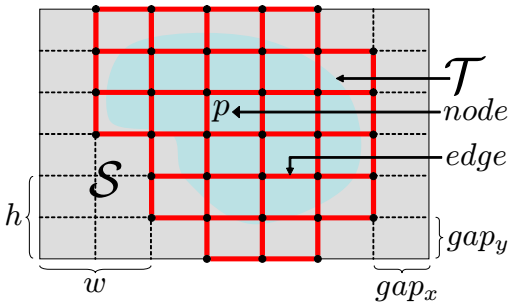


Fig. 4.3: The nodes and edges of an MRF associated with image completion. In this example, the w, h parameters were set equal to $w = 2gap_x$, $h = 2gap_y$.

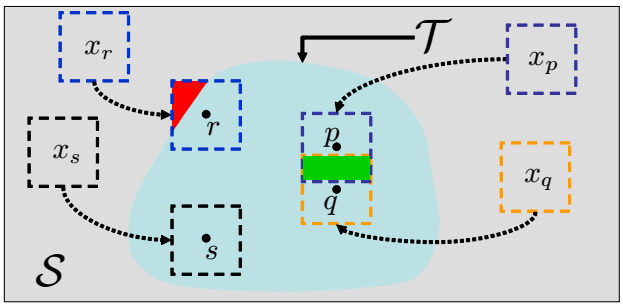


Fig. 4.4: For the boundary node r, its label cost $V_r(x_r)$ will be an SSD over the red region while for nodes p, q their potential $V_{pq}(x_p, x_q)$ will be an SSD over the green region. Node s is an interior node and so its label cost $V_s(x_s)$ will always be zero.

4.2 Image completion as a discrete global optimization problem

Given an input image $\mathcal{I}_0$ as well as a *target region* $\mathcal{T}$ and a *source region* $\mathcal{S}$ (where $\mathcal{S}$ is always a subset of $\mathcal{I}_0 - \mathcal{T}$), the goal of image completion is to fill $\mathcal{T}$ in a visually plausible way simply by copying patches from $\mathcal{S}$. We propose to turn this into a discrete optimization problem with a well defined objective function. To this end we propose the use of the following discrete Markov Random Field (MRF):

The labels $\mathcal{L}$ of the MRF will consist of all $w \times h$ patches from the source region $\mathcal{S}$[1]. For defining the nodes of the MRF an image lattice will be used with an horizontal and vertical spacing of gap_x and gap_y pixels respectively. The MRF nodes $\mathcal{V}$ will be all lattice points whose $w \times h$ neighborhood intersects the target region, while the edges $\mathcal{E}$ of the MRF will make up a 4-neighborhood system on that lattice (see Figure 4.3).

[1]Hereafter each label (*i.e* patch) will be represented by its center pixel

The single node potential $V_p(x_p)$ (called *label cost* hereafter) for placing patch x_p over node p will encode how well that patch agrees with the source region around p and will equal the following sum of squared differences (SSD):

$$V_p(x_p) = \sum_{dp \in [-\frac{w}{2} \ \frac{w}{2}] \times [-\frac{h}{2} \ \frac{h}{2}]} \mathcal{M}(p+dp)\big(\mathcal{I}_0(p+dp) - \mathcal{I}_0(x_p+dp)\big)^2, \tag{4.1}$$

where $\mathcal{M}(\cdot)$ is a binary mask non zero only in region $\mathcal{S}$ (since $\mathcal{M}(\cdot)$ is zero outside $\mathcal{S}$ the label costs of interior nodes, *i.e* nodes whose $w \times h$ neighborhood does not intersect $\mathcal{S}$, will obviously be all zero). In a similar fashion the pairwise potential $V_{pq}(x_p, x_q)$ due to placing patches x_p, x_q over neighbors p, q will measure how well these patches agree at the resulting region of overlap and will again be given by the SSD over that region (see Figure 4.4). Note that gap_x and gap_y are set so that such a region of overlap always exists.

Based on this formulation our goal will then be to assign a label $\hat{x}_p \in \mathcal{L}$ to each node p so that the total energy $\mathcal{F}(\hat{x})$ of the MRF is minimized where:

$$\mathcal{F}(\hat{x}) = \sum_{p \in \mathcal{V}} V_p(\hat{x}_p) + \sum_{(p,q) \in \mathcal{E}} V_{pq}(\hat{x}_p, \hat{x}_q). \tag{4.2}$$

Intuitively, any algorithm optimizing this energy is roughly solving a huge jigsaw puzzle where source patches are the puzzle pieces while region $\mathcal{T}$ represents the puzzle itself. One important advantage of our formulation is that it also provides a unified framework for texture synthesis and image inpainting. *E.g* to handle texture synthesis (where one wants to extend an input texture T_0 to a larger region T_1) one suffices to set $\mathcal{S} = T_0$ and $\mathcal{T} = T_1 - T_0$. Moreover, our framework allows the use of (what we call) *"completion by energy refinement"* techniques, one example of which we will see later.

4.3 Priority-BP

Furthermore, an additional advantage would be that we can now hopefully apply belief propagation (*i.e* a state-of-the-art optimization method) to our energy function. Unfortunately, however, this was not feasible. The reason was the intolerable computational cost of BP caused by the huge number of existing

labels. Motivated by this fact, one other major contribution of this work is the introduction of a novel MRF optimization scheme, called Priority-BP, that can deal exactly with this type of problems and carries two significant extensions over standard BP: one of them, called *dynamic label pruning*, is based on the key idea of drastically reducing the number of labels. However, instead of this happening beforehand (which will almost surely lead to throwing away useful labels) pruning takes place on the fly (*i.e* while BP is running) with a (possibly) different number of labels kept for each node. The important thing to note is that only the beliefs calculated by BP are used for that purpose. Furthermore, the second extension, called *priority-based message scheduling*, makes use of label pruning and allows us to always send cheap messages between the nodes of the graphical model. Moreover, it considerably improves BP's convergence thus accelerating completion even further.

The significance of our contribution also grows due to the fact that (as we shall see) Priority-BP is a generic algorithm applicable to any MRF energy function. This is unlike any prior use of Belief Propagation [53] and therefore, our method resolves for the first time what is currently considered one of the main limitations of BP: its inefficiency to handle problems with a huge number of labels. In fact this issue has been a highly active research topic over the last years. Until now, however, the techniques that have been proposed were valid only for restricted classes of MRFs [21, 45]. Not only that but our priority-based message scheduling scheme can be used (independently of label pruning) as a general method for accelerating the convergence of BP.

4.3.1 Priority-based message scheduling

BP is an iterative algorithm which works by propagating local messages along the nodes of an MRF [103]. Messages sent from node p to node q form a set $\{m_{pq}(x_q)\}_{x_q \in \mathcal{L}}$, where element $m_{pq}(x_q)$ indicates how likely node p thinks that node q should be assigned label x_q. Furthermore, messages are updated (*i.e* sent)

until convergence as follows[2]:

$$m_{pq}(x_q) = \min_{x_p \in \mathcal{L}} \Big\{ V_p(x_p) + V_{pq}(x_p, x_q) + \sum_{r: r \neq q, (r,p) \in \mathcal{E}} m_{rp}(x_p) \Big\} \tag{4.3}$$

After convergence, a set of beliefs $\{b_p(x_p)\}_{x_p \in \mathcal{L}}$ is computed for each node, where belief $b_p(x_p)$ is defined as follows:

$$b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p) \in \mathcal{E}} m_{rp}(x_p) \tag{4.4}$$

$b_p(x_p)$ approximates the max-marginal of the posterior at node p and is therefore roughly related to how likely label x_p is for that node. Based on this fact a node is then assigned the label of maximum belief i.e. $\hat{x}_p = \arg\max_{x_p \in \mathcal{L}} b_p(x_p)$. It is known that for tree structured graphs BP always gives the optimal solution while for graphs with loops if BP converges it can only guarantee to find a local optimum.

In this form, however, BP is impractical for problems with a large number of labels like ours. In particular, if $|\mathcal{L}|$ is the total number of labels (which, in our case, can be many many thousands) then just the basic operation of updating the messages from one node p to another node q takes $O(|\mathcal{L}|^2)$ time. In fact the situation is much more worse for us. The huge number of labels also implies that for any pair of adjacent nodes p, q their matrix of pairwise potentials $V_{pq}(\cdot, \cdot)$ is so large that cannot fit into memory and therefore be precomputed. That matrix therefore must be reestimated every time node p sends its messages to node q, meaning that $|\mathcal{L}|^2$ SSD calculations (between image patches) are needed for each such update.

To deal with this issue we will try to reduce the number of labels by exploiting the beliefs calculated by BP. However not all nodes have beliefs which are adequate for this purpose in our case. To see that it suffices to observe that the label costs at all interior nodes are all equal to zero. This in turn implies that the beliefs at an interior node will initially be all equal as well, meaning that the node is "unconfident" about which labels to prefer. No label pruning may therefore take place and so any message originating from that node will be very expensive to calculate, *i.e* it will take $O(|\mathcal{L}|)$ time. On the contrary, if we had a node whose

[2]We work in the $-log$ domain so we use the min-sum version of BP

labels could be pruned (and assuming that the maximum number of labels after pruning is L_{max} with $L_{max} \ll |\mathcal{L}|$), then any message from that node would take only $O(L_{max})$ time.

Based on this observation we therefore propose to use a specific message scheduling scheme whose goal will be twofold. On on hand, it will make label pruning possible and favor the circulation of cheap messages. On the other hand it will speed up BP's convergence. This issue of BP message scheduling, although known to be crucial for the success of BP, it has been largely overlooked until now. Also, to the best of the author's knowledge it is the first time that message scheduling is used in this manner for general graphical models. Roughly, our message scheduling scheme will be based on the notion of priorities that are assigned to the nodes of the MRF. Any such priority will represent a node's confidence about which labels to prefer and will be dynamically updated throughout the algorithm's execution. Our message scheduling will then obey the following simple principle:

Message-scheduling principle. *The node most confident about its labels should be the first one (i.e. it has the highest priority) to transmit outgoing messages to its neighbors.*

There are two reasons why one may want to do this. The first is that the more confident a node is, the more label pruning it can tolerate (before sending its outgoing messages) and therefore the cheaper these messages will be. The second reason is that we also help other nodes become more amenable to pruning this way. Intuitively, this happens because the more confident a node is the more informative its messages are going to be, meaning that these messages can help the neighbors of that node to increase their own confidence and thus become more tolerable to pruning as well. Furthermore, by first propagating the most informative messages around the graphical model we also help BP to converge much faster. This has been verified experimentally as well. *E.g* Priority-BP never needed more than a small fixed number of iterations to converge for all of our image completion examples.

A pseudocode description of Priority-BP is contained in algorithm 1. Each iteration of Priority-BP is divided into a forward and a backward pass. The actual message scheduling mechanism, as well as label pruning takes place during the

Algorithm 1 Priority-BP

assign priorities to nodes and declare them uncommitted
for $k = 1$ to K **do** {K is the number of iterations}
  execute *ForwardPass* and then *BackwardPass*
end for
assign to each node p its label $\hat{x}_p$ that maximizes $b_p(\cdot)$

ForwardPass:
for time = 1 to N **do** {N is the number of nodes}
  p = "uncommitted" node of highest priority
  apply "label pruning" to node p
  forwardOrder[time] = p ; $p\rightarrow$committed = true;
  for any "uncommitted" neighbor q of node p **do**
    send all messages $m_{pq}(\cdot)$ from node p to node q
    update beliefs $b_q(\cdot)$ as well as priority of node q
  end for
end for

BackwardPass:
for time = N to 1 **do**
  p = forwardOrder[time]; $p\rightarrow$committed = false;
  for any "committed" neighbor q of node p **do**
    send all messages $m_{pq}(\cdot)$ from node p to node q
    update beliefs $b_q(\cdot)$ as well as priority of node q
  end for
end for

forward pass. This is also where one half of the messages gets transmitted (*i.e* each MRF edge is traversed in only one of the 2 directions). To this end all nodes are visited in order of priority. Each time we visit a node, say p, we mark it as "committed" meaning that we must not visit him again during the current forward pass. We also prune its labels and then allow him to transmit its "cheap" (due to pruning) messages to all of its neighbors apart from the committed ones (as these have already sent a message to p during the current pass). The priorities of all neighbors that received a new message are then updated and the process continues with the next uncommitted (*i.e* unvisited) node of highest priority until no more uncommitted nodes exist.

The role of the backward pass is then just to ensure that the other half of the messages gets transmitted as well. To this end we do not make use of priorities, but simply visit the nodes in reverse order (with respect to the order of the forward pass) just transmitting the remaining unsent messages from each node. For this reason no label pruning takes place during this pass. We do update node

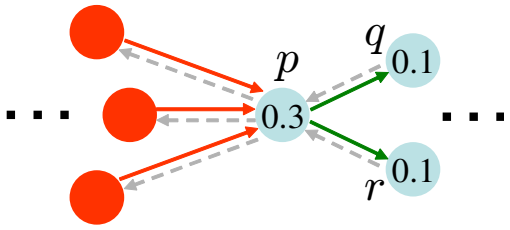


Fig. 4.5: Message scheduling during the forward pass: currently only red nodes have been committed and only messages on red edges have been transmitted. Among uncommitted nodes (*i.e* blue nodes) the one with the highest priority (*i.e* node p) will be committed next and will also send messages only along the green edges (*i.e* only to its uncommitted neighbors q, r). Messages along dashed edges will be transmitted during the backward pass. Priorities are indicated by the numbers inside uncommitted nodes.

priorities, though, so that they are available during the next forward pass.

Also, as we shall see, a node's priority depends only on the current beliefs at that node. One big advantage out of this is that keeping the node priorities up-to-date can be done very efficiently in this case, since only priorities for nodes with newly received messages need to be updated. The message scheduling mechanism is further illustrated in Figure 4.5.

4.3.2 Assigning priorities to nodes

It is obvious that our definition of priority will play a very crucial role for the success of the algorithm. As already mentioned priority must relate to how confident a node is about the labels that should be assigned to him with the more confident nodes having higher priority. An important thing to note in our case is that the confidence of a node will depend solely on information that will be extracted by the BP algorithm itself. This makes our algorithm generic (*i.e* applicable to any MRF energy function) and therefore appropriate for a very wide class of problems.

In particular our definition of confidence (and therefore priority as well) for node p will depend only on the current set of beliefs $\{b_p(x_p)\}_{x_p \in \mathcal{L}}$ that have been estimated by the BP algorithm for that node. Based on the observation that belief $b_p(x_p)$ is roughly related to how likely label x_p is for node p, one way to measure the confidence of this node is simply by counting the number of likely labels, *e.g* those whose belief exceed a certain threshold b_{conf}. The intuition for this is that the greater this number the more labels with high probability exist for that node

and therefore the less confident that node turns out to be about which specific label to choose. And vice versa, if this number is small then node p needs to choose its label only among a small set of likely labels. Of course only relative beliefs $b_p^{rel}(x_p) = b_p(x_p) - b_p^{max}$ (where $b_p^{max} = \max_{x_p \in \mathcal{L}} b_p(x_p)$) matter in this case and so by defining the set $\mathbb{CS}(p) = |\{x_p \in \mathcal{L} : b_p^{rel}(x_p) \geq b_{conf}\}|$ (which we will call the *confusion set* of node p hereafter) the priority of p is then inversely related to the cardinality of that set:

$$\text{priority}(p) = \frac{1}{|\mathbb{CS}(p)|} \tag{4.5}$$

This definition of priority also justifies why during either the forward or the backward pass we were allowed to update priorities only for nodes that had just received new incoming messages: the reason is that the beliefs (and therefore the priority) of a node may change only if at least one incoming message to that node changes as well (this is true due to the way beliefs are defined *i.e* $b_p(x_p) = -V_p(x_p) - \sum_{r:(r,p)\in\mathcal{E}} m_{rp}(x_p)$). Although we tested other definitions of priority as well (e.g. by using an entropy-like measure on beliefs) the above criterion for quantifying confidence gave the best results in practice by far.

4.3.3 Applying Priority-BP to image completion

We pause here for a moment (postponing the description of label pruning to the next section) in order to stress the advantages of applying our algorithm to image completion while also showing related results.

First of all we should mention that although confidence has already been used for guiding image completion in other works as well [36, 40], our use of confidence differs (with respect to these approaches) in two very important aspects. The first is that we use confidence in order to decide upon the order of BP message passing and not for greedily deciding which patch to fill next. These are two completely different things: the former is part of a principled global optimization procedure, while the latter just results in patches that cannot change their appearance after they have been filled.

The second aspect is that in all of the previous approaches the definition of confidence was mostly based either on heuristics or on ad hoc principles that were simply making use of application-specific knowledge about the image completion

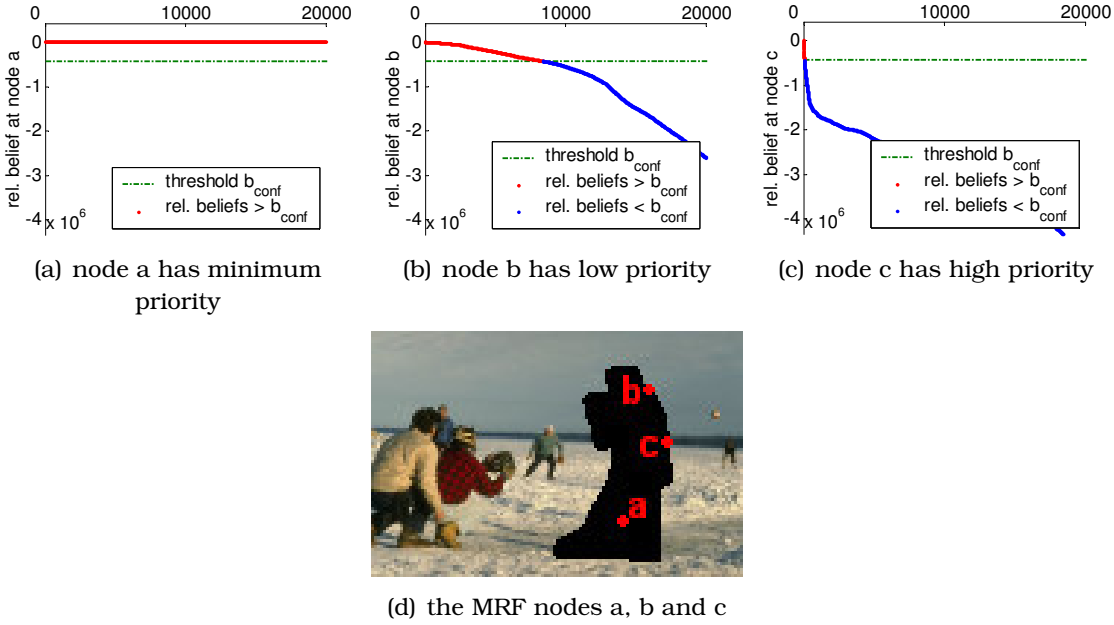
(a) original image (b) masked image (c) visiting order during first forward pass (d) Priority-BP result (e) result of [36]

Fig. 4.6: In column (c) darker patches correspond to nodes that are visited earlier during message scheduling at the first forward pass

process. On the contrary, as we saw, our definition of confidence is generic and therefore applicable to any kind of images. Moreover, this way our method is placed on firm theoretical grounds.

Three examples of applying Priority-BP to image completion are shown in Figure 4.6. As can be seen the algorithm has managed to fill the missing regions in a visually plausible way. The third column in that figure shows the visiting order of the nodes during the first forward pass (based on our definition of priority). The darker a patch is in these images, the earlier the corresponding node was visited. Notice how the algorithm learns by itself how to propagate first the messages of the nodes containing salient structure where the notion of saliency depends on each specific case. *E.g* the nodes that are considered salient for the first example of Figure 4.6 are those lying along the horizon boundary. On the contrary for the second example of that figure the algorithm prefers to propagate information along the MRF edges at the interior of the wooden trunk first. *The remarkable thing is that in both cases such information was not explicitly encoded, but was, instead, inferred by the algorithm.*

This is in contrast to the state-of-the-art method in [36] where the authors had to hardwire isophote-related information into the definition of priority (*i.e* a



(a) node a has minimum priority

(b) node b has low priority

(c) node c has high priority

(d) the MRF nodes a, b and c

Fig. 4.7: The plots in (a), (b) and (c) show the sorted relative beliefs for the MRF nodes a, b and c in figure (d) at the start of Priority-BP. Relative beliefs plotted in red correspond to labels in the confusion set. This set determines the priority of the corresponding node.

measure which is not always reliably extracted or even appropriate *e.g* in images with texture). The corresponding results produced by that method are shown in the last column of Figure 4.6. In these cases only one label (*i.e* patch) is greedily assigned to each missing block of pixels and so any errors made early cannot be later backtracked, thus leading to the observed visual inconsistencies. On the contrary, due to our global optimization approach, any errors that are made during the very first iterations can be very well corrected later, since our algorithm always maintain not one but many possible labels for each MRF node. A characteristic case for this is the third example in Figure 4.6, where unless one employs a global optimization scheme it is not easy to infer the missing structure.

Also, the plots in Figures 4.7(a), 4.7(b), 4.7(c) illustrate our definition of priority in (4.5). They display the largest 20000 relative beliefs (sorted in ascending order) that are observed at the very beginning of the algorithm for each of the MRF nodes a, b, c in Figure 4.7(d) respectively. Relative beliefs plotted in red correspond to labels in the confusion set. Node a, being an interior node, has initially all the labels in its confusion set (since their relative beliefs are all zero) and is therefore of lowest priority. Node b still has too many labels in its confusion set due to the uniform appearance of the source region around that node. On the contrary node

c is one of the nodes to be visited early during the first forward pass, since only very few labels belong to its confusion set. Indeed even at the very beginning we can easily exclude (*i.e* prune) many source patches from being labels of that node without the risk of throwing away useful labels. This is why Priority-BP prefers to visit him early.

4.3.4 Label pruning

The main idea of "label pruning" is that as we are visiting the nodes of the MRF during the forward pass (in the order induced by their priorities), we dynamically reduce the number of possible labels for each node by discarding labels that are unlikely to be assigned to that node. In particular, after committing a node, say p, all labels having a very low relative belief at p, say less than b_{prune}, are not considered as candidate labels for p thereafter. The remaining labels are called the *"active labels"* for that node. An additional advantage we gain this way is that after all MRF nodes have pruned their labels at least once (e.g. at the end of the first forward pass) then we can precompute the reduced matrices of pairwise potentials (which can now fit into memory) and thus greatly enhance the speed of our algorithm. The important thing to note is that "label pruning" relies only on information carried by the Priority-BP algorithm itself as well. This keeps our method generic and therefore applicable to any energy function. A key observation, however, relates to the fact that label pruning is a technique not meant to be used on its own. Its use is allowed only in conjunction with our priority-based message scheduling scheme of visiting most confident nodes first (*i.e* nodes for which label pruning is safe and does not throw away useful labels). This is exactly the reason why label pruning does not take place during the backward pass.

In practice we apply label pruning only to nodes whose number of active labels exceeds a user specified number L_{max}. To this end when we are about to commit a node we traverse its labels in order of belief (from high to low) and each such label is declared active until either no more labels with relative belief greater than b_{prune} exist or the maximum number of active labels L_{max} has been reached. In the case of image completion, however, it turns out that we also have to apply an additional filtering procedure as part of label pruning. The problem is that otherwise we may

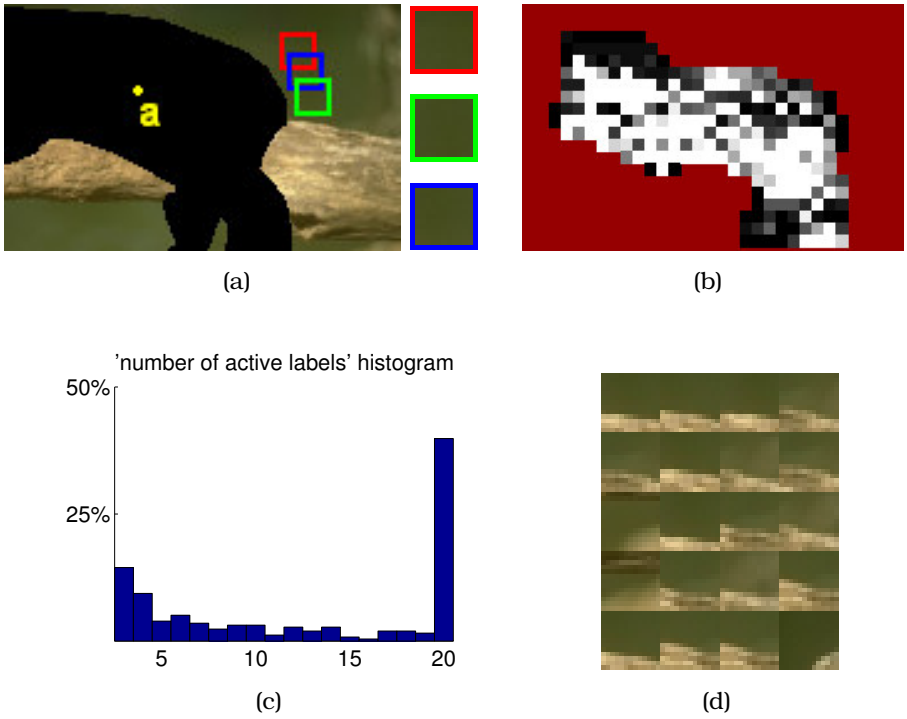


Fig. 4.8: (a) Although the red, green and blue patches correspond to distinct labels, they are very similar and so only one has to be an active label for a node. **(b)** A map with the number of active labels per node (for the 2^{nd} example of Figure 4.6). Darker patches correspond to nodes with fewer labels. As can be seen interior nodes often require more labels. **(c)** The corresponding histogram showing the percentage of nodes using a certain number (in the range $L_{min} = 3$ to $L_{max} = 20$) of active labels. **(d)** The active labels for node a in Fig. (a).

end up having too many active labels which are similar to each other, thus wasting part of the L_{max} labels we are allowed to use. This issue is further illustrated in Figure 4.8(a). To this end as we traverse the sorted labels we declare a label as active only if it is not similar to any of the already active labels (where similarity is measured by calculating the SSD between image patches), otherwise we skip that label and go to the next one. Alternatively, we apply a clustering procedure to the patches of all labels beforehand (*e.g* cluster them into textons) and then never use more than one label from each cluster while traversing the sorted labels. Finally, we should note that for all nodes a (user-specified) minimum number of active labels L_{min} is always kept.

The net result of label pruning is thus to obtain a compact and diverse set of active labels for each MRF node (all of them having reasonably good beliefs). *E.g* Figure 4.8(b) displays the number of active labels used by each of the nodes in the second example of Figure 4.6. The darker a patch is in that figure the fewer

are the active labels of the corresponding node. As it was expected, interior nodes often require more active labels to use. The corresponding histogram showing the percentage of nodes that use a certain number of active labels is displayed in Figure 4.8(c). Notice that more than half of the MRF nodes do not use the maximum number of active labels (which was $L_{max} = 20$ in this case). Also, Fig. 4.8(d) displays the active labels that have been selected by the algorithm for node a in Fig. 4.8(a).

4.4 Extensions & further results

Completion via energy refinement: One advantage of posing image completion as an optimization problem is that one can now refine completion simply by refining the energy function (*i.e* adding more terms to it). *E.g* to favor spatial coherence during image completion (*i.e* fill the target region with large chunks of the source region) one simply needs to add the following *"incoherence penalty terms"* V^0_{pq} to our energy function: $V^0_{pq}(x_p, x_q) = w_0$ if $x_p - x_q \neq p - q$, while in all other cases $V^0_{pq}(x_p, x_q) = 0$. These terms simply penalize (with a weight w_0) the assignment of non-adjacent patches (with centers x_p, x_q) to adjacent nodes p, q and have proved useful in texture synthesis problems (*e.g* see Figure 4.13). Thanks to the ability of Priority-BP to handle effectively any energy function we intend to explore the utility (with respect to image completion) of many other refinement terms in the future. We believe that this will also be an easy and effective way of applying prior knowledge or imposing user specified constraints on the image completion process.

Pyramid-based image completion: Another important advantage of our method is that it can also be used in multi-scale image completion, where a Gaussian pyramid of images $\mathcal{I}_k, \mathcal{I}_{k-1}, \ldots, \mathcal{I}_0$ is given as input. For this we begin by applying Priority-BP to the image at the coarsest scale $\mathcal{I}_k$. The output of this procedure is then up-sampled and the result, say $\mathcal{I'}_k$, is used for guiding the completion of the image $\mathcal{I}_{k-1}$ at the next finer scale. To this end the only part of our algorithm that needs to be modified is that of how label costs are computed. In particular the mask $\mathcal{M}$ in (4.1) will be now non zero everywhere and so not only pixels from the source region of $\mathcal{I}_{k-1}$ are taken into account, but also pixels from the

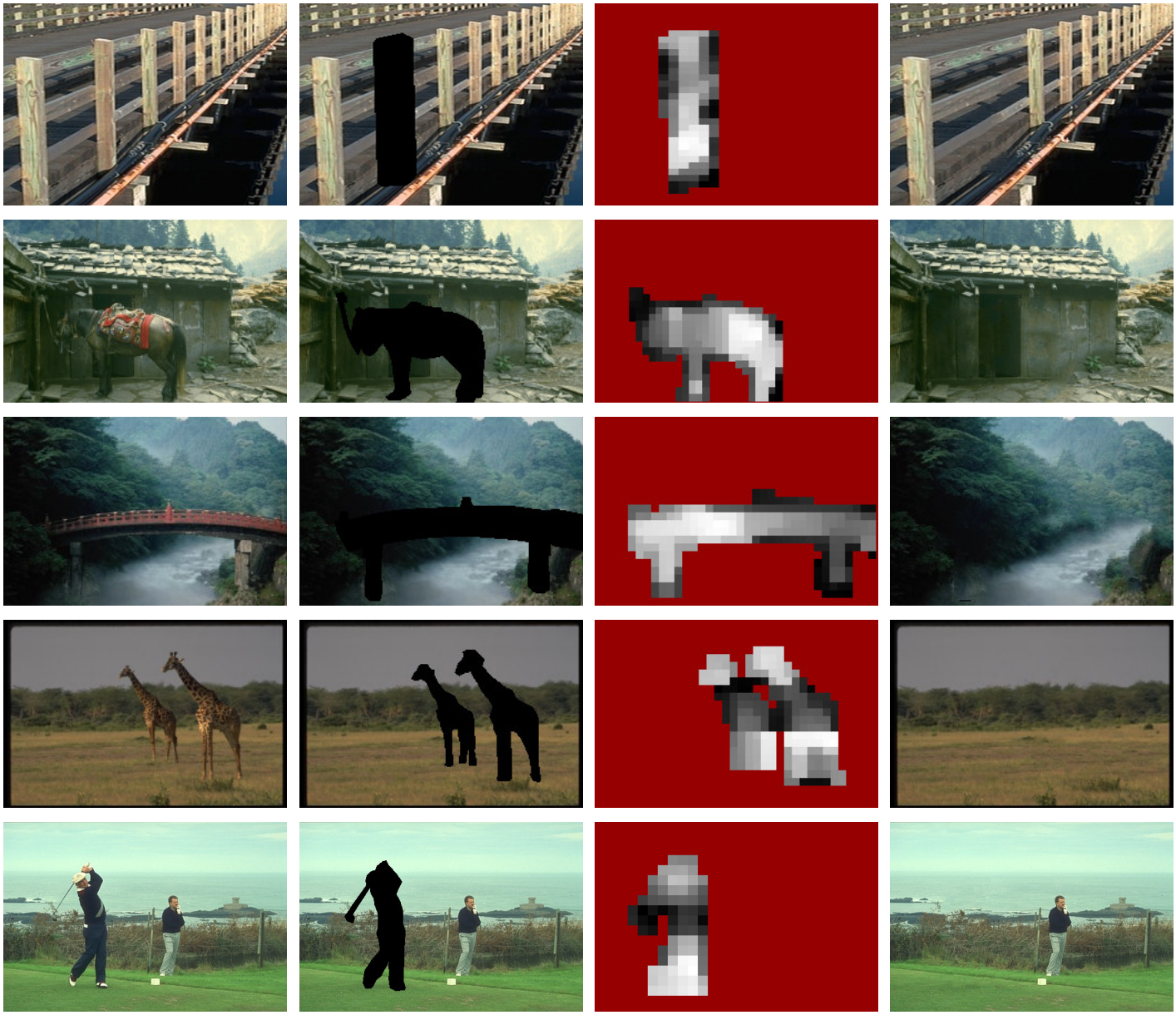
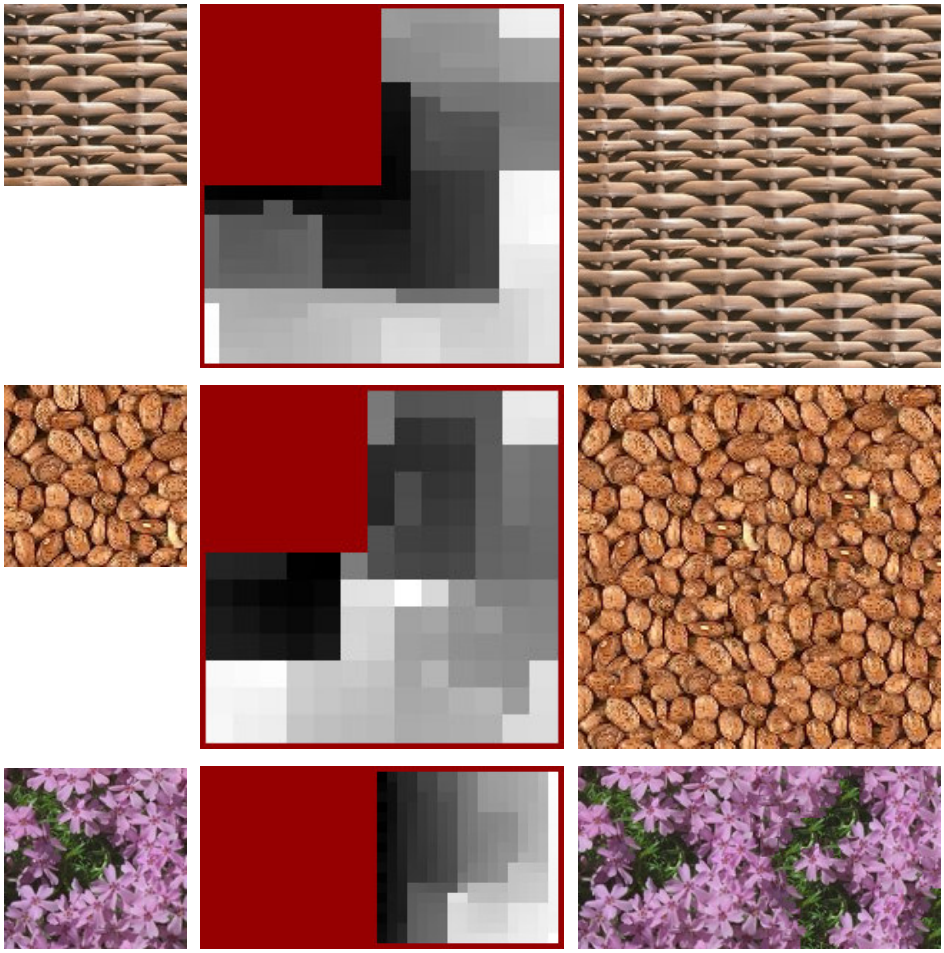


Fig. 4.9: Image completion. From left to right: original images, masked images, visiting order at 1^{st} forward pass, Priority-BP results

unknown target region, where now the values for these pixels are borrowed from the approximation image $\mathcal{I}'_k$. The rest of the algorithm remains the same and this process is repeated until we reach the image at the finest scale $\mathcal{I}_0$. An advantage we gain this way is that features at multiple scales can be captured.

Figures 4.9, 4.14 contain further results on image completion. These results along with those in Figure 4.6 demonstrate the effectiveness of our method, which was tested on a wide variety of input images. As can be seen from the presented examples, Priority-BP was able to handle the completion of smooth



(a) Input texture (b) Visiting order during 1^{st} forward pass (c) Output texture

Fig. 4.10: texture synthesis results produced with the Priority-BP algorithm

regions, textured areas, areas with structure as well as any combinations of the above. Also, figure 4.10 contains some examples on texture synthesis. These were again produced by utilizing our exemplar-based framework. In addition, Figure 4.11 demonstrates another possible application of Priority-BP. In this case, it has been used for accomplishing the task of removing text from images whereas, in Figure 4.12, Priority-BP has been employed as an image inpainting tool for the restoration of a destroyed digital photograph. Our method had no problem of handling these tasks as well. At this point, it is important to emphasize the fact that, in all of the above cases, exactly the same algorithm has been used.

In Figure 4.13 we demonstrate an example of using the "incoherence penalty terms" in texture synthesis. As one can observe the output texture does contain large chunks of the input texture as intended. Also, in the last example of Fig-

(a) Input image with text (b) Visiting order at 1^{st} forward pass (c) Priority-BP result

Fig. 4.11: An example of text removal

(a) Input image (b) Visiting order at 1^{st} forward pass (c) Priority-BP result

Fig. 4.12: An image inpainting example

ure 4.9 we show the final result of a pyramid-based image completion. In this case the input image was 481×321 and a 2-level pyramid has been used. We note here that, for all of the presented examples, the visiting order of the nodes during the first forward pass is shown as well. This contributes to illustrating how the algorithm initially chooses to propagate information (*i.e* messages) for each one of the input images. In our tests the patch size ranged between 7×7 and 27×27. The running time on a 2.4GHz CPU varied from a few seconds up to 2 minutes for

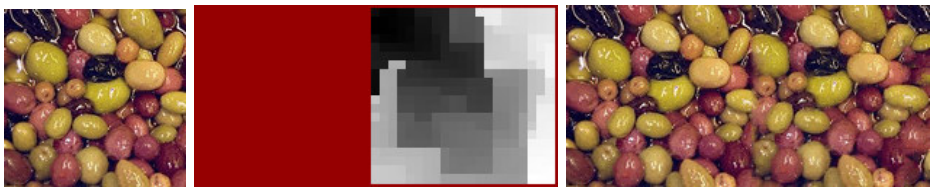


Fig. 4.13: Texture synthesis using the "incoherence penalty terms". Notice that, in this case, the output texture has been synthesized by copying large chunks from the input texture.

256× 170 images, while the maximum number of labels L_{max} was set between 10 and 50 (depending on the the input's difficulty). For all of the examples the belief thresholds were set equal to b_{conf}=−SSD$_0$, b_{prune}=−2·SSD$_0$, where SSD$_0$ represents a predefined mediocre SSD score between $w \times h$ patches. For the composition of the final patches these are usually blended with weights that are proportional to the confidence of the corresponding nodes. This technique gave very good results in practice, while more elaborate schemes like feathering or multi-resolution splining have been also tried in some cases.

Finally, another point, that is worth mentioning (as it brings a great reduction in the computational time), is the use of the fast Fourier Transform for performing all SSD calculations needed by the algorithm [72, 121]. More specifically, the estimation of all label costs as well as all pairwise potentials requires many SSD computations. *E.g*, as indicated by equation (4.1), for estimating the label costs for a node p we need to calculate the SSD between a local neighborhood around p, say $\mathcal{I}_0(p+dp)$, and every other source patch, say $\mathcal{I}_0(x_p+dp)$, with the result being multiplied by a mask $\mathcal{M}(p+dp)$, i.e.:

$$V_p(x_p) = \sum_{dp} \mathcal{M}(p+dp)\big(\mathcal{I}_0(p+dp) - \mathcal{I}_0(x_p+dp)\big)^2. \tag{4.6}$$

By defining, however, the following identities:

$$t \triangleq x_p$$
$$\mathcal{I}_1(dp) \triangleq \mathcal{I}_0(p+dp)$$
$$\mathcal{M}_1(dp) \triangleq \mathcal{M}(p+dp)$$

and substituting them into equation (4.6), that equation reduces to:

$$\begin{aligned} V_p(t) &= \sum_{dp} \mathcal{M}_1(dp)\big(\mathcal{I}_1(dp) - \mathcal{I}_0(t+dp)\big)^2 \\ &= \sum_{dp} \mathcal{M}_1(dp)\mathcal{I}_1(dp)^2 - 2\sum_{dp} \mathcal{M}_1(dp)\mathcal{I}_1(dp)\mathcal{I}_0(t+dp) + \sum_{dp} \mathcal{M}_1(dp)\mathcal{I}_0(t+dp)^2 \end{aligned}$$

which can be also written as an expression involving correlations between functions:

$$V_p(t) = \langle \mathcal{M}_1, \mathcal{I}_1^2 \rangle(0) - 2\langle \mathcal{M}_1 \cdot \mathcal{I}_1, \mathcal{I}_0 \rangle(t) + \langle \mathcal{M}_1, \mathcal{I}_0^2 \rangle(t).$$

In the above expression, $\langle \cdot , \cdot \rangle$ denotes the correlation operation. Furthermore, the first term (i.e. $\langle \mathcal{M}_1, \mathcal{I}_1^2 \rangle(0)$) is independent of t, which means that it can be precomputed, and so we can estimate all values of function $V_p(\cdot)$ just by using two correlation operations. However, the important thing to note is that these correlations can now be computed very efficiently simply by moving to the frequency domain and using the fast Fourier Transform (FFT) [109] therein. This way of performing the computations greatly accelerates the whole process and can be applied for estimating the pairwise potentials $V_{pq}(\cdot, \cdot)$ as well.

4.5 Conclusions

A novel approach which treats image completion, texture synthesis and image inpainting in a unified manner has been presented. To avoid visually inconsistent results due to greedy patch assignments, we pose all of these tasks in the form of a discrete labeling problem with a well defined objective function. To solve that problem a novel global optimization scheme, Priority-BP, has been proposed that carries two very important extensions over standard BP: priority-based message scheduling and label pruning. Our algorithm does not rely on image-specific prior knowledge and can be applied to any kind of images. Furthermore, it is generic (*i.e* applicable to any MRF energy) and thus copes with one of the main limitations of BP: its inefficiency to handle problems with a huge number of labels. Finally, a wide variety of examples have verified its effectiveness.

Fig. 4.14: Some more results on image completion, produced using the Priority-BP algorithm. From left to right: original images, masked images, visiting order at 1^{st} forward pass, Priority-BP results

CHAPTER 5

3D Visual Reconstruction of Large Scale Natural Sites

During this chapter, as an application of our LP-based MRF optimization techniques that we have introduced earlier, we will turn our attention to a different research topic: the proposal of novel image based modeling and rendering methods, which are capable of automatically reproducing faithful (i.e. photorealistic) digital copies of complex 3D virtual environments, while also allowing the virtual exploration of these environments at interactive frame rates. This topic lies at the convergence of two research fields that are normally considered to be on opposite sides: computer vision and computer graphics. Traditionally, computer graphics starts with input geometric models and tries to produce images, while computer vision works the other way around, i.e. it starts with images (or image sequences) and produces geometric models. Recently, however, there has been a convergence of these two fields somewhere in the middle and the main reason for that was the introduction of the so-called IBMR (Image Based Modeling and Rendering) techniques. In this case, computer graphics is concerned with the image-based-rendering part, while computer vision is employed during the image-based-modeling process of IBMR methods. IBMR techniques have received growing interest over the last years and many of the related techniques that have been proposed managed to give excellent results in a lot of cases. Nevertheless, despite the many advantages of IBMR methods, these techniques also exhibit certain limitations. For instance, in order to work properly, they typically require very large amount of image data, which has as a result that these methods are difficult to use for the virtual reconstruction of large scale 3D environments. Thus, despite the large number of IBMR methods that have been proposed so far, only few of them are capable of dealing with 3D scenes of large size.

Based on these observations, and in order to overcome the limitations of current IBMR methods, this chapter presents a hybrid (geometry- & image-based) IBMR

technique suitable for providing interactive walkthroughs of large, complex outdoor scenes. To this end, a new hybrid representation of a 3D scene is proposed, called "morphable 3D-mosaics". In our case, motion is restricted along a smooth predefined path and the input to our system is a sparse set of stereoscopic views at certain points (key-positions) along that path (one view per position). An approximate local 3D model is constructed from each view, capable of capturing photometric and geometric properties of the scene only locally. Then during the rendering process, a continuous morphing (both photometric & geometric) takes place between successive local 3D models, using what we call a "morphable 3D-model". The morphing proceeds in a physically-valid way. For this reason, a wide-baseline image matching technique is proposed, handling cases where the wide baseline between the two images is mainly due to a looming of the camera. Our system can be also extended in the event of multiple stereoscopic views (and therefore multiple local models) per key-position of the path (related by a camera rotation). In that case one local 3D-mosaic (per key-position) is constructed comprising all local 3D models therein and a "morphable 3D-mosaic" is used during the rendering process. A partial-differential equation is adopted to handle the problem of geometric consistency of each 3D-mosaic.

As we shall see, both for the local 3D-models extraction, as well as for the construction of the morphable 3D-mosaics (i.e. the morphing estimation), MRFs will play a very crucial role. Therefore, robust as well as efficient techniques are needed for optimizing them. As expected, our MRF optimization methods introduced in chapter 3 are going to be used for that purpose. Based on these observations, the main goals of this chapter are thus twofold: On one hand, it presents the main ingredients of our IBMR framework and explains its main contributions. On the other hand, it also shows how our MRF techniques can be employed as part of a complete practical application. In addition, we should note that our framework has already been successfully applied for the virtual reconstruction of the Samaria gorge in Crete, which is one of the largest and most magnificent gorges in Europe. Therefore, for demonstrating the effectiveness of our framework, a sample from the results that were obtained during this virtual reconstruction of the Samaria gorge will be presented at the end of this chapter as well.

Reality is merely an illusion, albeit a very persistent one.

—Albert Einstein (1879-1955)

5.1 Introduction

One research problem of computer graphics that has attracted a lot of attention over the last years is the creation of modeling and rendering systems capable to provide photorealistic & interactive walkthroughs of complex, real-world environments. Two are the main approaches that have been proposed so far for that purpose. On one hand, there exist those techniques that are geometry-based. techniques first try to estimate an accurate global 3D model of the scene. They then use the extracted 3D model in order to render the scene under any given viewpoint. One of their advantages is that they provide great flexibility and allow many of the scene's properties to be modified during rendering. E.g. by having a global 3D model one can readily alter not only the viewpoint, but also the lighting conditions of the scene. However, their big disadvantage comes from the fact that extracting an accurate global 3D model can be either extremely time consuming or very difficult (not to say impossible) in many cases. For example such a 3D-model construction task can be easy for scenes containing mostly planar objects (e.g. architectural-type scenes), but becomes extremely hard for outdoor scenes containing objects with irregular geometry e.g. trees. The automatic extraction of a 3D-model from images, also known as multiple view geometry, has been (and still is) an active research topic in computer vision. In fact a significant amount of progress has been achieved in this area over the last years [44, 60, 92].

A second class of techniques that has emerged during the last years are the so-called Image Based Rendering (IBR) methods [96]. These techniques concentrate their effort directly on how to fill the pixels of a novel view and skip the geometric modeling of the scene completely. In place of the geometric modeling, a dense set of images inside the scene is captured as a first step. One then tries to synthesize any given view by appropriately resampling the previously acquired set of captured images. By thinking of the world's appearance as a dense array of light rays filling the space, one can easily see that what all image based rendering methods

actually try to do is to reconstruct the so-called *plenoptic function* [1]. This is a 7-dimensional function $P(V_x, V_y, V_z, \theta, \phi, \lambda, t)$, which models a 3D dynamic environement by recording the light rays at every space location (V_x, V_y, V_z), towards every possible direction (θ, ϕ), over any range of wavelengths λ and at any time t (see Figure 5.1). Each time we capture an image by a camera, the light rays passing through the camera's center of projection are recorded and so that image can be considered as a specific sample of the plenoptic function. Based on these observations, image based rendering can thus be thought of as the signal processing task of *reconstructing a continuous functions (in this case the plenoptic function) based only on a discrete set of samples from that function.* As IBR methods make use of actual images from the scene under consideration, one of their greatest advantages is the fact that they can attain high levels of photorealism. However, this comes at the price of requiring a big number of captured images. This actually forms one of the biggest problem of IBR methods and is the main reason that the great majority of existing IBR techniques can be applied only to scenes of either small or medium scale. If one tries to apply such techniques to large scale scenes, then he is confronted with a huge amount of data required which makes these methods impractical for such cases.

So, while a lot of research has been done regarding small scale scenes, there are only few examples of work dealing with large scale environments. The presented framework is such an example of a hybrid (geometry and image based) approach, capable of providing photorealistic and interactive walkthroughs of large-scale, complex outdoor environments. To this end, one major contribution of this work is the proposal of a novel data representation for a 3D scene, called *morphable 3D-mosaics*, consisting of a series of morphable (both geometrically and photometrically) 3D models. The main assumption is that during the walkthrough, the user motion takes place along a (relatively) smooth, predefined path

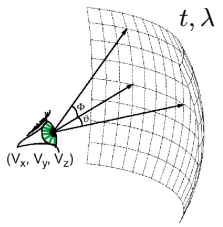


Fig. 5.1: A schematic view of the plenoptic function

of the environment. The input to our system is then a sparse set of stereoscopic views captured at certain locations (which we will call *"key-positions"* hereafter) along that path (see Figure 5.2a). Assuming that initially there is only one view per key-position, a series of *local 3D models* are then constructed, one for each stereoscopic view, with these local models capturing the photometric and geometric properties of the scene at a local level and containing only an approximate representation of the scene's geometry (see Figure 5.2b). Then, instead of trying to create a global 3D model out of all these local models (a task that can prove to be extremely difficult in many cases and requires a very accurate registration between local models), we rather follow a different approach. The key idea is that during the transition between any two successive key-positions pos_1, pos_2, along the path (with corresponding local models L_1 and L_2), a *"morphable 3D-model"* L_{morph} is displayed by the rendering process (see Figure 5.2c). At point pos_1 this model coincides with L_1, while as we are approaching pos_2 it is gradually transformed into L_2, coinciding with the latter upon reaching key-position pos_2. It is important to note that this morphing between local models is both photometric and geometric. Moreover, we always ensure that the morphing proceeds in a physically-valid way and is thus transparent to the user of the system. To this end, a wide-baseline image matching technique is proposed which is capable of extracting a dense field of correspondences between images whose difference in appearance is mainly due to a looming of the camera. Therefore, during the rendering process, and as the user traverses the predefined path, a continuous morphing between successive local 3D models takes place all the time.

Our system can be also extended to handle the existence of multiple stereoscopic views per key position of the path, which are all related by a pure rotation of

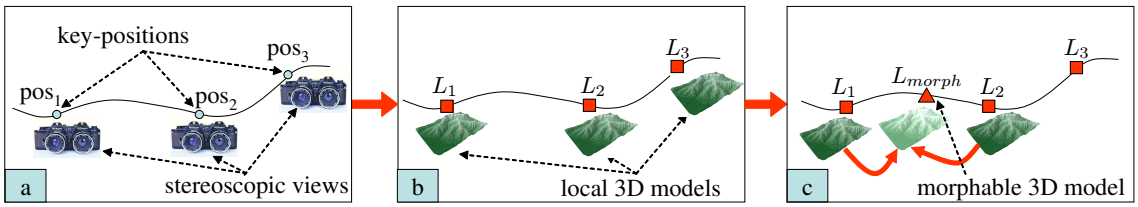


Fig. 5.2: Overview of our approach: **(a)** A sparse set of stereoscopic views is captured at key-positions along the path **(b)** One local 3D model is constructed out of each stereoscopic view **(c)** As the user traverses the path a morphable 3D model is displayed during rendering. This way a continuous morphing between successive local models takes place at any time, with this morphing being both photometric as well as geometric.

the stereoscopic camera. In that case, there will also be multiple local models per key-position. Therefore, before applying the morphing procedure, a *3D-mosaic* per key-position needs to be constructed as well. Each 3D-mosaic will simply comprise the multiple local models at the corresponding key-position and will itself be a bigger local model covering a wider field of view. Morphing can then proceed in the same way as before with the only difference being that these 3D-mosaics will be the new local 3D models to be used during the stage of morphing (in place of the smaller individual ones). So, during morphing, instead of a morphable 3D model we will now have a *morphable 3D mosaic*.

Regarding the advantages of the proposed framework, the following points could be made:

- To start with, it offers a very flexible way of representing a 3D scene in the sense that one can put more emphasis either on the geometric or the image based representation of the scene simply by respectively decreasing or increasing the number of key-positions.

- No global 3D model of the environment needs to be assembled, a process which can be extremely cumbersome and error-prone for large scale scenes e.g. the global registration of multiple local models can accumulate a great amount of error, while it also presumes a very accurate extraction of the underlying geometry. On the contrary, no such accurate geometric reconstruction of the individual local 3D models nor a very precise registration between them is required by our framework in order that it can produce satisfactory results.

- On the other hand, by making use of an image-based data representation, our framework is also capable of reproducing the photorealistic richness of the scene.

- At the same time it offers scalability to large scale environments, as only one "morphable 3D-model" is displayed at any time, while it also makes use of a rendering path which is highly optimized in modern 3D graphics hardware.

- Data acquisition is very easy (e.g. collecting the stereoscopic images for a path over 100 meters long took us only about 20 minutes) and requires no

special or expensive equipment (just a pair of digital cameras and a tripod)

- Finally, our framework makes up an end-to-end system thus providing an almost automated processing of the input data which are just a sparse set of stereoscopic images.

Besides proposing a novel data representation for a 3D scene that makes use of a concurrent photometric and geometric morphing procedure, various other contributions are included as part of our image-based modeling and rendering system:

- For instance, in the context of photometric morphing, a robust method for obtaining a dense field of correspondences between wide baseline images is proposed. On one hand, this task is reduced to a discrete energy minimization problem. On the other hand, for dealing with the existence of a wide baseline, the resulting change of scale between pixels is also taken into account during matching.

- Similarly, as part of obtaining a physically valid geometric morphing, a novel approach for extracting the required 3D correspondences between local 3D models is included. Our method is based on solving just a standard partial differential equation and is very fast.

- Furthermore, in the context of the 3D mosaics construction, a technique for combining local 3D models (related to each other by a 3D rotation) is presented, which is again based on solving a standard partial differential equation. Our method can cope with errors in the geometry of the local 3D models and always ensures that a consistent 3D mosaic is generated. To this end, geometric rectifications are applied to each one of the local 3D models during their merging.

- Finally, as part of our rendering pipeline, we propose the use of modern graphics hardware to perform both photometric and geometric morphing, thus drastically reducing the rendering time.

5.2 Related work

Many examples of geometry-based modeling methods of real world scenes can be found in the computer vision literature [76, 99, 120, 123–125, 131]. One such characteristic example is the work of Pollefey et al. [106] on 3D reconstruction from hand-held cameras. Debevec et al. [37,38] propose a hybrid (geometry- and image-based) approach, which makes use of view dependent texture mapping. However, their work is mostly suitable for architectural type scenes. Furthermore, they also assume that a basic geometric model of the whole scene can be recovered interactively. In [50], an image-based technique is proposed by which an end-user can create walkthroughs from a sequence of photographs, while in "plenoptic modeling" [97] a warp operation is introduced that maps panoramic images (along with disparity) to any desired view. However, this operation is not very suitable for use in modern 3D graphics hardware. Lightfield [86] and Lumigraph [57] are two popular image-based rendering methods, but they require a large number of input images and so they are mainly used for small scale scenes.

To address this issue work on unstructured/sparse lumigraphs has been proposed by various authors. One such example is the work of Buehler et al. [29]. However, in that work, a fixed geometric proxy (which is supposed to describe the global geometry of the scene at any time instance) is being assumed, an assumption that is not adequate for the case of 3D data coming from a sequence of sparse stereoscopic views. This is in contrast to our work where view-dependent geometry is being used due to the continuous geometric morphing that is taking place. Another example of a sparse lumigraph is the work of Schirmacher et al. [116]. Although they allow the use of multiple depth maps, any possible inconsistencies between them are not taken into account during rendering. This is again in contrast to our work, where an optical flow between wide-baseline images is estimated to deal with this issue. Furthermore, this estimation of optical flow between wide baseline images reduces the required number of views. For these reasons, if any of the above two approaches were to be applied to large-scale scenes, like those handled in our case, many more images (than ours) would then be needed. Also, due to our rendering path which can be highly optimized in modern graphics hardware, we can achieve very high frame rates during rendering, while the corresponding frame rates listed in [116] are much lower due to

an expensive barycentric coordinate computation which increases the rendering time.

In [137] Vedula et al. make use of a geometric morphing procedure as well, but it is used for a different purpose which is the recovery of the continuous 3D motion of a non-rigid dynamic event (e.g. human motion). Their method (like some other methods [95, 152]) uses multiple synchronized video streams combined with IBR techniques to render a dynamic scene, but all of these approaches are mostly suitable for scenes of smaller scale (than the ones we are interested in), since they assume that all of the cameras are static. Also, in the "Interactive visual tours" approach [135], video (from multiple cameras) is being recorded as one moves along predefined paths inside a real world environment and then image based rendering techniques are used for replaying the tour and allowing the user to move along those paths. This way virtual walkthroughs of large scenes can be generated. Finally, in the "sea of images" approach [3], a set of omnidirectional images are captured for creating interactive walkthroughs of large, indoor environments. However, this set of images is very dense with the image spacing being ≈ 1.5 inches.

5.3 Overview of the modeling pipeline

A diagram of our system's modeling pipeline is shown in Figure 5.3. We will first consider the simpler case of having only one stereoscopic view per key-position of the path.

Prior to capturing these stereoscopic views, a calibration of the stereoscopic camera needs to take place first. During this stage both the external parameters (i.e. the relative 3D rotation and translation between the left and right camera),

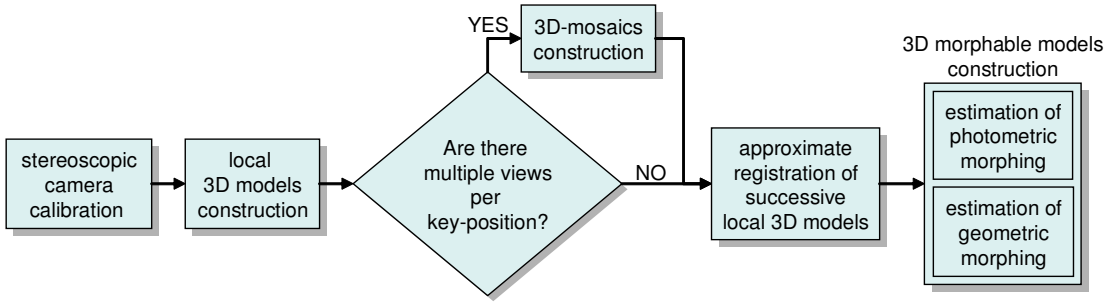


Fig. 5.3: The modeling pipeline

as well as the internal parameters of the stereoscopic camera are estimated. We make the common assumption that both the left and right camera are modeled by the usual pinhole. In this case their internal parameters are contained in the so-called intrinsic matrices K_{left}, K_{right}. Any such matrix has the following form:

$$\begin{bmatrix} f_x & c & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where (f_x, f_y) represents the focal length, c describes the skewness of the 2 image axes while (u_0, v_0) represents the principal point. We also model (both radial and tangential) lens distortion and the following model is assumed for this purpose:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} 1 + d_1 r^2 + d_2 r^4 + d_5 r^6 + 2d_3 xy + d_4(r^2 + 2x^2) \\ 1 + d_1 r^2 + d_2 r^4 + d_5 r^6 + d_3(r^2 + 2y^2) + 2d_4 xy \end{bmatrix}$$

where (x, y) are the ideal (distortion-free) pixel coordinates, $(\hat{x}, \hat{y})$ are the corresponding observed image coordinates and $r = \sqrt{x^2 + y^2}$. For estimating all of these parameters we apply a method similar to that in [23], using as input stereoscopic image pairs of a calibrated chess pattern captured at random positions and orientations by our camera (see Figure 5.4).

After the camera calibration has finished, then the following stages of the modeling pipeline need to take place:

1. *Local 3D models construction (section 5.4):* A photometric and geometric representation of the scene near each key-position of the path is constructed. The geometric part of a local model needs to be only an approximation of the true scene geometry.

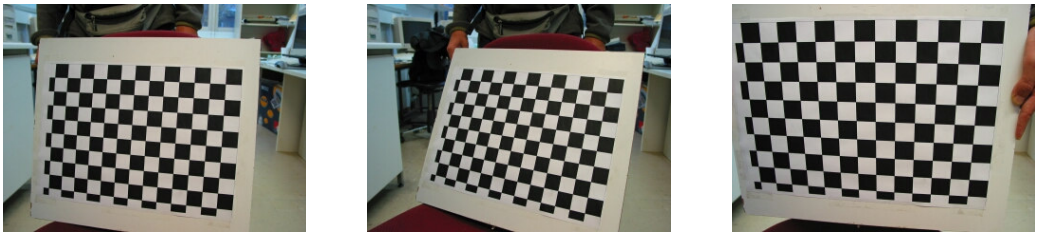


Fig. 5.4: For calibrating our camera we capture images of a chess pattern at random positions and orientations.

2. *Approximate registration between successive local 3D models (section 5.5):* An estimation of the relative pose between successive local models takes place here. We should note that only a coarse estimate of the relative pose is needed, since this will not be used for an exact registration of the local models, but merely for the morphing procedure that takes place later.

3. *3D morphable models construction (section 5.6):* The photometric as well as the geometric morphing between successive local 3D models is estimated during this stage of the modeling pipeline.

In the case that there are multiple views per key position of the path, then, as already explained, there will also have to be an additional stage responsible for the *3D-mosaics construction*. This stage needs to take place prior to the registration step and is described in section 5.8. Finally, we describe the rendering pipeline of our system in section 5.7.

5.4 Local 3D models construction

For each stereoscopic image pair, a 3D model describing the scene locally (i.e. as seen from the camera viewpoint) must be produced during this stage. To this end, a stereo matching procedure is applied to the left and right images (denoted I_{left} and I_{right}), so that disparity can be estimated for all points inside a selected image region dom_0 of I_{left} (see section 5.4.1 about how this disparity can be estimated). Using then the resulting disparity map (as well as the calibration matrices of the cameras) a 3D reconstruction takes place and thus the maps X_0, Y_0 and Z_0 are produced (see Fig. 5.5(a)). These maps respectively contain the x, y and z coordinates of the reconstructed points with respect to the 3D coordinate system of the left camera.

The set $L_0 = (X_0, Y_0, Z_0, I_{left}, dom_0)$ consisting of the images X_0, Y_0, Z_0 (the geometric-maps), the image region dom_0 (valid domain of geometric-maps) and the image I_{left} (the photometric map) makes up what we call a *"local model"* L_0. Hereafter that term will implicitly refer to such a set of elements. By applying a 2D triangulation on the image grid of a local model, a textured 3D triangle mesh can be produced. The 3D coordinates of triangle vertices are obtained from the underlying geometric maps while texture is obtained from I_{left} and mapped onto

(a) (b)

Fig. 5.5: (a) Depth map Z_0 of a local model (black pixels do not belong to its valid region dom_0). **(b)** A rendered view of the local model using an underlying triangle mesh

the mesh (see Fig. 5.5(b)). It should be noted that the geometric maps of a local model are expected to contain only an approximation of the scene's true geometric model.

5.4.1 Disparity estimation

Disparity estimation proceeds in two stages (see Figure 5.6). During the first stage, we reduce the problem of stereo matching to a discrete labeling problem which is going to be solved through the energy optimization a 1^{st} order Markov Random Field. The nodes of the corresponding MRF are going to be the pixels of the left image and the single node potential for assigning disparity d_p to pixel p is going to be estimated as follows:

$$V_p(d_p) = |I_{right}(p - d_p) - I_{left}(p)|^2$$

Furthermore, for the pairwise potentials the truncated semimetric distance between disparities has been used, i.e.:

$$V_{pq}(d_p, d_q) = \min(\lambda_0, |d_p - d_q|^2) \ ,$$

where λ_0 denotes the maximum allowed penalty that can be imposed. For the optimization of the above MRF, the LP-based algorithms (introduced in chapter 3) have been used, since they can always guarantee a solution which is close to the optimal one. The role of the first stage is to produce a good initial estimate of the disparity and to avoid any bad local minima during the optimization process.

Its output is then given as input to the next stage of the disparity estimation

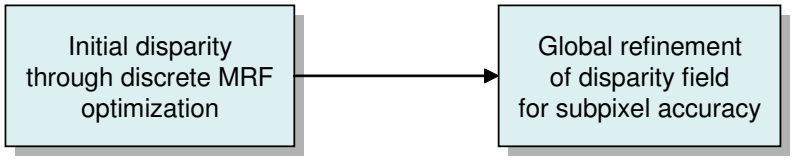


Fig. 5.6: The 2 stages needed for disparity estimation

process, where a global refinement of the disparity field is taking place. To this end, the energy of a first order Markov Random Field is again being minimized. The difference, however, with respect to the first stage, is that now a local continuous optimization scheme is being used so that disparities with subpixel accuracy can be obtained. In particular, we use a standard gradient descent type algorithm for minimizing the following energy function:

$$\sum_{(i,j)} \big(I_{right}(i - d_{ij}, j) - I_{left}(i,j)\big)^2 + \lambda \sum_{(i,j)} \sum_{p \in N_{ij}} g(d_{ij} - d_p) \ ,$$

where N_{ij} is the 4-point neighborhood of pixel (i, j) and d_{ij} again represents the unknown disparity field. The λ parameter is a regularization parameter, while the potential function $g(\cdot)$ is chosen to be discontinuity adaptive [19] (e.g. a truncated quadratic distance), so that a regularized solution, which also preserves discontinuities, is finally computed. The disparity field is initialized with the values estimated during the first stage. Due to this initialization the gradient descent algorithm usually converges very fast and does not get trapped to any poor local minima.

5.5 Relative pose estimation between successive local models

Let L_k=(X_k,Y_k,Z_k, I_k, dom_k) and L_{k+1}=(X_{k+1},Y_{k+1}, Z_{k+1}, I_{k+1}, dom_{k+1}) be 2 successive local models along the path. For their relative pose estimation, we need to extract a set of point matches (p_i, q_i) between the left images I_k, I_{k+1} of models L_k, L_{k+1} respectively (see section 5.5.1). Assuming that such a set of matches already exists, then the pose estimation can proceed as follows: the 3D points of L_k corresponding to p_i are $P_i = (X_k(p_i), Y_k(p_i), Z_k(p_i))$ and so the reprojections of p_i on image I_{k+1} are: $p'_i = K_{left}(R \cdot P_i + T) \in \mathbb{P}^2$, where R (a

3×3 orthonormal matrix) and T (a 3D vector) represent the unknown rotation and translation respectively.

So the pose estimation can be achieved by minimizing the following reprojection error:

$$\sum_i dist(q_i, p_i')^2$$

where $dist$ denotes euclidean image distance. For this purpose, an iterative constrained-minimization algorithm may be applied with rotation represented internally by a quaternion q ($\|q\| = 1$). The *essential matrix* (also computable by the help of the matches (p_i, q_i) and K_{left}, K_{right}) can be used to provide an initial estimate [60] for the iterative algorithm.

5.5.1 Wide-baseline feature matching under camera looming

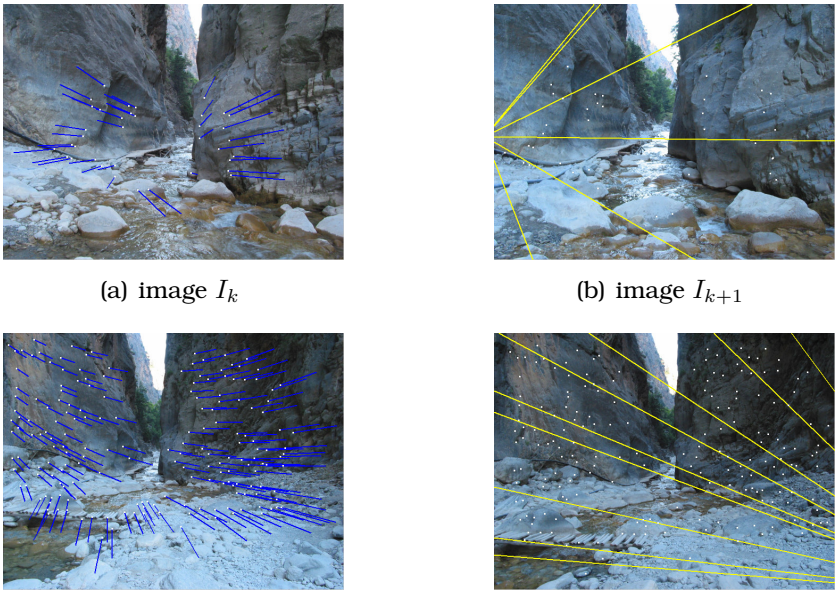
Therefore the pose estimation problem is reduced to that of extracting a sparse set of correspondences between I_k, I_{k+1}. A usual method for tackling the latter problem is the following: first a set of interest-points in I_k are extracted (using an interest-point detector). Then for each interest-point, say p, a set of candidate points CAND$_p$ inside a large rectangular region SEARCH$_p$ of I_{k+1} are examined and the best one is selected according to a similarity measure. Usually the candidate points are extracted by applying an interest-point detector to region SEARCH$_p$ as well.

However unlike left/right images of a stereoscopic view, I_k and I_{k+1} are separated by a wide baseline. Simple measures like correlation have been proved extremely inefficient in such cases. Assuming a smooth predefined path (and therefore a smooth change in orientation between I_k, I_{k+1}), it is safe to assume that the main difference at an object's appearance in images I_k and I_{k+1}, comes from the forward camera motion along the Z axis (looming). The idea for extracting valid correspondences is then based on the following observation: the dominant effect of an object being closer to the camera in image I_{k+1} is that its image region in I_{k+1} appears scaled by a certain scale factor $s>1$. That is, if $p \in I_k, q \in I_{k+1}$ are corresponding pixels: $I_{k+1}(sq) \approx I_k(p)$. So an image patch of I_k at p should look similar to an image patch of an appropriately rescaled (by s^{-1}) version of I_{k+1}.

Of course, the scale factor s varies across the image. Therefore the following

(a) (b)

Fig. 5.7: (a) Image I_k along with computed optical flow vectors (blue segments) for all points marked white. **(b)** Image I_{k+1} along with matching points (also marked white) for all marked points of (a). A few epipolar lines are also shown. In both images, the yellow square around a point is analogous to the point's estimated scale factor (10 scales $S = \{1, 0.9^{-1}, ..., 0.1^{-1}\}$ have been used).



(a) image I_k (b) image I_{k+1}

(c) image I_k (d) image I_{k+1}

Fig. 5.8: Two more examples (one example per row) of wide baseline matching from another scene. Optical flow vectors (on image I_k) as well as estimated epipolar lines (on image I_{k+1}) are shown again. Also, notice the large camera motion taking place in the top example e.g. the stones in the water appear much closer to the camera in figure (b) than in figure (a).

strategy, for extracting reliable matches, can be applied:

1. Quantize the scale space of s to a discrete set of values $S = \{s_j\}_{j=0}^{n}$, where $1 = s_0 < s_1 < ... < s_n$

2. Rescale I_{k+1} by the inverse scale s_j^{-1} for all $s_j \in S$ to get rescaled images I_{k+1,s_j}

   For any $q \in I_{k+1}$, $p \in I_k$, let us denote by $\overline{I_{k+1,s_j}(q)}$ a (small) fixed-size patch

around the projection of q on I_{k+1,s_j} and by $\overline{I_k(p)}$ an equal-size patch of I_k at p.

3. Given any point $p \in I_k$ and its set of candidate points $\text{CAND}_p = \{q_i\}$ in I_{k+1}, use correlation to find among the patches at any q_i and across any scale s_j, the one most similar to the patch of I_k at p:

$$(q', s') = \arg\max_{q_i, s_j} corr(\ \overline{I_{k+1,s_j}(q_i)},\ \overline{I_k(p)}\ ).$$

This way, apart from a matching point $q' \in I_{k+1}$, a scale estimate s' is provided for point p as well.

The above strategy has been proved very effective, giving a high percentage of exact matches even in cases with very large looming. Such an example can be seen in Fig. 5.7 wherein the images baseline is ≈ 15 meters, resulting in scale factors of size ≈ 2.5 for certain image regions. Even if we set as candidate points CAND_p of a point p, all points inside SEARCH_p in the other image (and not only detected interest-points therein), the above procedure still picks the right matches in most cases. The results in Figure 5.8 have been produced in this way.

5.6 Morphing estimation between successive local models

At the current stage of the modeling pipeline, a series of approximate local 3D models (along with approximate estimates of the relative pose between every successive two) are available to us. Let $L_k = (X_k, Y_k, Z_k, I_k, dom_k)$, $L_{k+1} = (X_{k+1}, Y_{k+1}, Z_{k+1}, I_{k+1}, dom_{k+1})$ be such a pair of successive local models and pos_k, pos_{k+1} their corresponding key-positions on the path. By making use of the approximate pose estimate between L_k and L_{k+1}, we will assume hereafter that the 3D vertices of both models are expressed in a common 3D coordinate system.

Rather than trying to create a consistent global model by combining all local ones (a rather tedious task requiring among others high quality geometry and pose estimation) we will instead follow a different approach, which is based on

the following observation: near path point pos_k, model L_k is ideal for representing the surrounding scene. On the other hand, as we move forward along the path approaching key-position of the next model L_{k+1}, the photometric and geometric properties of the environment are much better captured by that model. (For example compare the fine details of the rocks that are revealed in Fig. 5.7(b) and are not visible in Fig. 5.7(a)). So during transition from pos_k to pos_{k+1}, we will try to gradually morph model L_k into a new destination model, which should coincide with L_{k+1} upon reaching point pos_{k+1}. (In fact, only part of this destination model can coincide with L_{k+1} since in general L_k, L_{k+1} will not represent exactly the same part of the scene). This morphing should be geometric as well as photometric (the latter wherever possible) and should proceed in a physically valid way. For this reason, we will use what we call a *"morphable 3D-model"*:

$$L_{morph} = L_k \cup (X_{dst}, Y_{dst}, Z_{dst}, I_{dst})$$

In addition to including the elements of L_k, L_{morph} also consists of maps $X_{dst}, Y_{dst}, Z_{dst}$ and map I_{dst} containing respectively the destination 3D vertices and destination color values for all points of L_k. At any time during the rendering process, the 3D coordinates $vert_{ij}$ and color col_{ij} of the vertex of L_{morph} at point (i, j) will then be:

$$vert_{ij} = \begin{bmatrix} (1-m)X_k(i,j) + mX_{dst}(i,j) \\ (1-m)Y_k(i,j) + mY_{dst}(i,j) \\ (1-m)Z_k(i,j) + mZ_{dst}(i,j) \end{bmatrix} \tag{5.1}$$

$$col_{ij} = (1-m)I_k(i,j) + mI_{dst}(i,j) \tag{5.2}$$

where m is a parameter determining the amount of morphing ($m=0$ at pos_k, $m=1$ at pos_{k+1} and $0<m<1$ in between). Specifying therefore L_{morph} amounts to filling-in the values of the destination maps $\{X, Y, Z, I\}_{dst}$ for each point $p \in dom_k$.

For this purpose, a 2-step procedure will be followed that depends on whether point p has a physically corresponding point in L_{k+1} or not:

1. Let Ψ be that subset of region $dom_k \subseteq I_k$, consisting only of those L_k points that have physically corresponding points in model L_{k+1} and let $u_{k \to k+1}$ be a function which maps these points to their counterparts in the I_{k+1} image. (Region Ψ represents that part of the scene which is common to both models

L_k, L_{k+1}). Since model L_k (after morphing) should coincide with L_{k+1}, it must then hold:

$$\begin{bmatrix} X_{dst}(p) \\ Y_{dst}(p) \\ Z_{dst}(p) \\ I_{dst}(p) \end{bmatrix} = \begin{bmatrix} X_{k+1}(u_{k\to k+1}(p)) \\ Y_{k+1}(u_{k\to k+1}(p)) \\ Z_{k+1}(u_{k\to k+1}(p)) \\ I_{k+1}(u_{k\to k+1}(p)) \end{bmatrix} \ \forall p \in \Psi \tag{5.3}$$

Points of region Ψ are therefore transformed both photometrically and geometrically.

2. The rest of the points (that is points in $\bar{\Psi}{=}dom_k \backslash \Psi$) do not have counterparts in model L_{k+1}. So these points will retain their color value (from model L_k) at the destination maps and no photometric morphing will take place:

$$I_{dst}(p) = I_k(p), \ \forall p \in \bar{\Psi} \tag{5.4}$$

But we still need to apply geometric morphing to those points so that no distortion/discontinuity in the 3D structure is observed during transition from pos_k to pos_{k+1}. Therefore we still need to fill-in the destination 3D coordinates for all points in $\bar{\Psi}$.

The 2 important remaining issues (which also constitute the core of the morphing procedure) are:

- How to compute the mapping $u_{k\to k+1}$. This is equivalent to estimating a 2D optical flow field between the left images I_k and I_{k+1}.
- And how to obtain the values of the destination geometric-maps at the points inside region $\bar{\Psi}$, needed for the geometric morphing therein.

Both of these issues will be the subject of the two subsections that follow.

5.6.1 Estimating optical flow between wide-baseline images I_k and I_{k+1}

In general, obtaining a reliable, relatively-dense optical flow field between wide-baseline images like I_k and I_{k+1} is a particularly difficult problem. Without addi-

tional input, usually only a sparse set of optical flow vectors can be obtained in the best case. In this case the basic problems are:

1. For every point in I_k, a large region of image I_{k+1} has to be searched for obtaining a corresponding point. This way the chance of an erroneous optical flow vector increases significantly (as well as the computational cost)

2. Simple measures (like correlation) are very inefficient for comparing pixel blocks between wide-baseline images

3. Even if both of the above problems are solved, optical flow estimation is inherently an ill-posed problem and additional assumptions are needed. In particular, we need to somehow impose the condition that the optical flow field will be piecewise smooth.

For dealing with the first problem, we will make use of the underlying geometric maps X_k,Y_k,Z_k of model L_k as well as the relative pose between I_k and I_{k+1}. By using these quantites, we can theoretically reproject any point, say p, of I_k onto image I_{k+1}. In practice since all of the above quantities are estimated only approximately, this permits us just to restrict the searching over a smaller region R_p around the reprojection point. The search region can be restricted further by taking the intersection of R_p with a small zone around the epipolar line corresponding to p. In addition, since we are interested in searching only for points of I_{k+1} that belong to dom_{k+1} (this is where L_{k+1} is defined), the final search region SEARCH$_p$ of p will be $R_p \cap dom_{k+1}$. If SEARCH$_p$ is empty, then no optical flow vector will be estimated and point p will be considered as not belonging to region Ψ.

For dealing with the second problem, we will use a technique similar to the one described in section 5.5.1 for getting a sparse set of correspondences. As already stated therein, the dominant effect due to a looming of the camera is that pixel neighborhoods in image I_{k+1} are scaled by a factor varying across the image. The solution proposed therein was to compare image patches of I_k not only with patches from I_{k+1}, but also with patches from rescaled versions of the latter image. We will use the same technique here, with the only difference being that instead of doing that for a sparse group of features we will now apply it to a dense set of pixels of image I_k. For this purpose we will again use a discrete set

of scale factors $S = \{1{=}s_0{<}s_1{<}...{<}s_n\}$ and we will rescale image I_{k+1} by each one of these factors where, as before, image I_{k+1} rescaled by s^{-1} (with $s \in S$) will be denoted by $I_{k+1,s}$. As we shall see in the next paragraph, this will have the effect of having to change the type of labels that we will use in the associated labeling problem.

Finally, to deal with the ill-posed character of the problem, we will first reduce the optical flow estimation to a discrete labeling problem and then formulate it in terms of minimizing the energy of a first order Markov Random Field [87]. What is worth noting here is that, contrary to a standard optical flow estimation procedure, the labels will now consist of vectors $l = (d_x, d_y, s) \in \mathbb{R}^2 \times S$, where the first 2 coordinates denote the components of the optical flow vector while the third one denotes the scale factor. This means that after labeling, not only an optical flow, but also a scale estimation will be provided for each point (see Fig. 5.10(a)). Given a label l, we will denote its optical flow vector by $flow(l) = (d_x, d_y)$ and its scale by $scale(l) = s$. Based on what was already mentioned above, the labels which are allowed to be assigned to a point p in I_k will be coming from the following set: $\text{LABELS}_p = \{q - p : q \in \text{SEARCH}_p\} \times S$. This definition of the label set LABELS_p simply encodes the following two things:

- For any point p of the first image, we are searching for corresponding points q only inside the restricted region SEARCH_p
- We also search across all scales in S, i.e. given a candidate matching point $q \in \text{SEARCH}_p$ for p, we compare patch $\overline{I_k(p)} \in I_k$ with any of the patches $\overline{I_{k+1,s}(q)} \in I_{k+1,s}$ where the scale s traverses all the elements of set S (see Figure 5.9). As before $\overline{I_k(p)}$ denotes a fixed size patch around p, while $\overline{I_{k+1,s}(q)}$ denotes an equal-size patch, which is located around the projection of q on the rescaled image $I_{k+1,s}$.

Getting an optical flow field is then equivalent to picking one element from the cartesian product $\text{LABELS} = \prod_{p \in \Psi} \text{LABELS}_p$. In our case, that element x of LABELS, which minimizes the following energy should be chosen:

$$\mathcal{F}(x) = \sum_{(p,p') \in \aleph} V_{pp'}(x_p, x_{p'}) + \sum_{p \in \Psi} V_p(x_p)$$

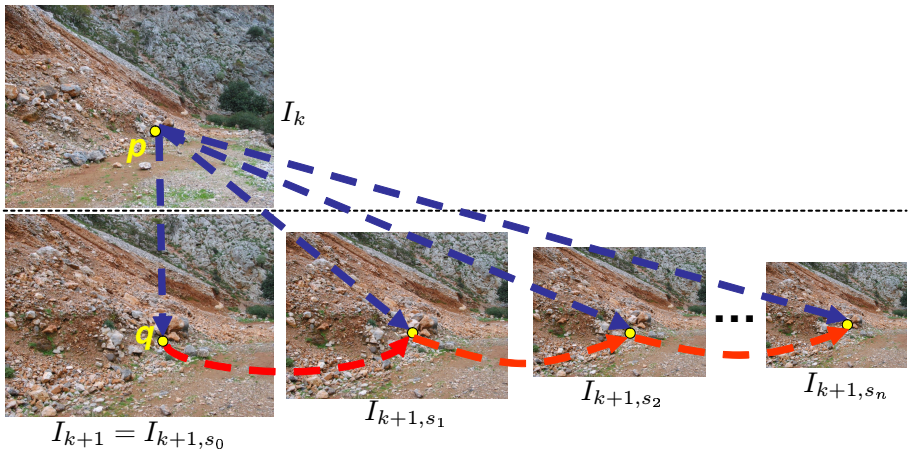
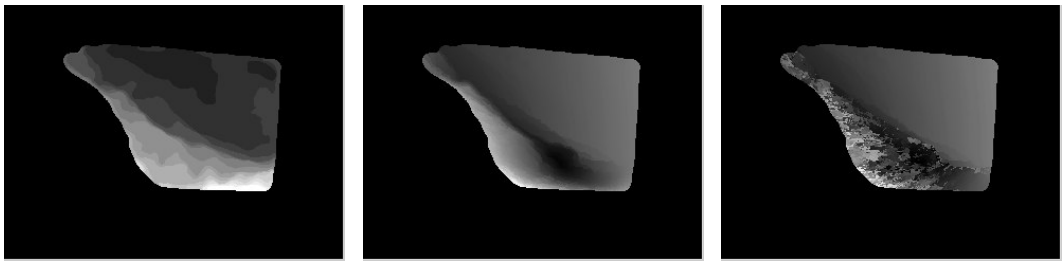


Fig. 5.9: Given a point $p \in I_k$ and a candidate matching point $q \in I_{k+1}$, we search across a range of scales $1 = s_0 < s_1 < ... < s_n$ by first projecting q on rescaled images and then comparing the neighborhood of each of the resulting pixels with the neighborhood of p in I_k.



(a) (b) (c)

Fig. 5.10: Maps of: **(a)** *scale factors* and **(b)** *optical flow magnitudes* for all points in Ψ, as estimated after applying the optical flow algorithm to the images of Fig. 5.7 and while using 10 possible scales $S = \{1, 0.9^{-1}, ..., 0.1^{-1}\}$. **(c)** Corresponding optical flow magnitudes when only one scale $S = \{1\}$ has been used. As expected, in this case the algorithm fails to produce exact optical flow for points that actually have larger scale factors. We note that darker pixels in a grayscale image correspond to smaller values.

The first sum in $\mathcal{F}(x)$ represents the prior term and penalizes optical flow fields which are not piecewise smooth, while the second sum in the above energy represents the likelihood and measures how well the corresponding optical flow agrees with the observed image data. The symbol $\aleph$ denotes a set of interacting pairs of pixels inside Ψ (we typically assume a 4-system neighborhood) and $V_{pp'}(\cdot,\cdot)$ denotes the pairwise potential function of the MRF. In our case, this function can be set as follows:

$$V_{pp'}(x_p, x_{p'}) = \min\big(||flow(x_p) - flow(x_{p'})||^2 + |scale(x_p) - scale(x_{p'})|^2, \lambda_0\big) \ ,$$

where λ_0 denotes the maximum pairwise penalty that can be imposed. Simpler pairwise potential functions, like the Potts function, have been also tested.

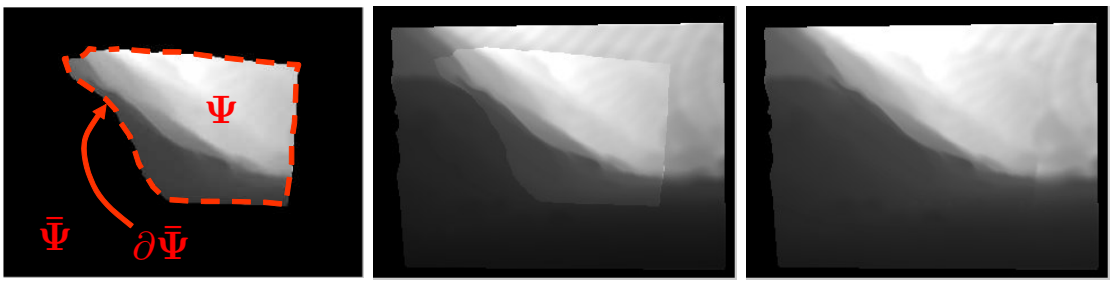
Regarding the terms $V_p(x_p)$, these measure the correlation between corresponding image patches as determined by the labeling x. According to a labeling x, for a point p in I_k its corresponding point is the projection into image $I_{k+1,scale(x_p)}$ of point $p + flow(x_p)$. This means that we should compare the patches $\overline{I_k(p)}$ and $\overline{I_{k+1,scale(x_p)}(p + flow(x_p))}$ and, for this reason, we set:

$$V_p(x_p) = corr(\ \overline{I_k(p)}\ ,\ \overline{I_{k+1,scale(x_p)}(p + flow(x_p))}\ )$$

The above energy $\mathcal{F}(x)$ can be minimized using any of the LP-based MRF optimization algorithms that we have introduced during chapter 3 of this thesis. The resulting optical flow, obtained when using the two images of Figure 5.7 as input, is shown in Figure 5.10. For comparison, we also show there (Figure 5.10(c)) the corresponding optical flow result, which is estimated if no search across scales takes place i.e. $S = \{1\}$. As expected, in this case, the resulting optical flow is very noisy for regions that are actually undergoing a large change of scale.

5.6.2 Geometric morphing in region $\bar{\Psi}$

After estimation of optical flow $u_{k\rightarrow k+1}$, we may apply equation (5.3) to all points in Ψ and thus fill the arrays $X_{dst}, Y_{dst}, Z_{dst}$ therein (see Fig. 5.11(a)). Therefore, at



(a) (b) (c)

Fig. 5.11: (a) Destination depth map Z_{dst} for points inside region Ψ after using optical flow of Fig. 5.10(b) and applying eq. (5.3). To completely specify morphing we need to extend this map to the points in region $\bar{\Psi}$ **(b)** Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ without applying geometric morphing. Notice that there exist discontinuities along the boundary $\partial\bar{\Psi}$. **(c)** Depth map Z_{dst} of (a) extended to points in $\bar{\Psi}$ after applying geometric morphing.

(a) (b) (c)

Fig. 5.12: Rendered views of the morphable 3D-model during transition from the key-position corresponding to image 5.7(a) to the key-position of image 5.7(b): **(a)** when no geometric morphing is applied to points in $\bar{\Psi}$ and **(b)** when geometric morphing is applied to points in $\bar{\Psi}$. **(c)** A close-up view of the rendered image in (b). Although there is no geometric discontinuity, there is a difference in texture resolution between the left part of the image (points in $\bar{\Psi}$) and the right part (points in Ψ) because only points of the latter part are morphed photometrically.

this stage of the modeling pipeline, the values of the destination geometric maps $X_{dst}, Y_{dst}, Z_{dst}$ are known for all points inside region Ψ, but are unknown for all points inside region $\bar{\Psi} = dom_k \backslash \Psi$ (i.e. the region which is the complement of Ψ in dom_k). Hereafter, the already known values of the destination geometric maps will be denoted by $\hat{X}_{dst}, \hat{Y}_{dst}, \hat{Z}_{dst}$, i.e. we define:

$$\hat{X}_{dst} \equiv X_{dst}|_{\Psi},\ \hat{Y}_{dst} \equiv Y_{dst}|_{\Psi},\ \hat{Z}_{dst} \equiv Z_{dst}|_{\Psi}$$

To completely specify morphing, we still need to fill the values of the destination geometric maps for all points in $\bar{\Psi} = dom_k \backslash \Psi$. In other words, we need to specify the destination 3D vertices for all points of L_k in $\bar{\Psi}$. Since these points do not have a physically corresponding point in L_{k+1}, we cannot apply (5.3) to get a destination 3D vertex from model L_{k+1}. The simplest solution would be that no geometric morphing is applied to these points and that their destination vertices just coincide with their L_k vertices. However, in that case:

- points in Ψ will have destination vertices from L_{k+1},
- while points in $\bar{\Psi}$ will have destination vertices from L_k

The problem resulting out of this situation is that the produced destination maps $X_{dst}, Y_{dst}, Z_{dst}$ (see Figs. 5.11(b), 5.12(a)) will contain discontinuities along the boundary (say $\partial\bar{\Psi}$) between regions Ψ and $\bar{\Psi}$, causing this way annoying discontinuity artifacts (holes) in the geometry of the "morphable 3D-model" during

the morphing procedure. This will happen because the geometry of both L_k and L_{k+1}, as well as their relative pose, have been estimated only approximately, and therefore these two models may not match perfectly when placed in a common 3D coordinate system.

The right way to fill-in the destination vertices at the points in $\bar{\Psi}$ is based on the observation that a physically valid destination 3D model should satisfy the following 2 conditions:

1. On the boundary of $\bar{\Psi}$, no discontinuity in 3D structure should exist, i.e. the unknown values of $X_{dst}, Y_{dst}, Z_{dst}$ along the boundary $\partial\bar{\Psi}$ should match the corresponding known values specified by $\hat{X}_{dst}, \hat{Y}_{dst}, \hat{Z}_{dst}$ along that boundary.

2. In the interior of $\bar{\Psi}$, the relative 3D structure of the initial L_k model should be preserved.

Intuitively, these two conditions simply imply that, as a result of morphing, vertices of L_k inside $\bar{\Psi}$ must be deformed without distorting their relative 3D structure so as to seamlessly match the 3D vertices of L_{k+1} along the boundary of $\bar{\Psi}$. In mathematical terms the first condition obviously translates to:

$$X_{dst}|_{\partial\bar{\Psi}} = \hat{X}_{dst}|_{\partial\bar{\Psi}},\ Y_{dst}|_{\partial\bar{\Psi}} = \hat{Y}_{dst}|_{\partial\bar{\Psi}},\ Z_{dst}|_{\partial\bar{\Psi}} = \hat{Z}_{dst}|_{\partial\bar{\Psi}}$$

while the second condition, which imposes the restriction of preserving the relative 3D structure of L_k, simply implies:

$$\begin{bmatrix} X_{dst}(p) - X_{dst}(p') \\ Y_{dst}(p) - Y_{dst}(p') \\ Z_{dst}(p) - Z_{dst}(p') \end{bmatrix} = \begin{bmatrix} X_k(p) - X_k(p') \\ Y_k(p) - Y_k(p') \\ Z_k(p) - Z_k(p') \end{bmatrix}, \forall p, p' \in \bar{\Psi}$$

which is easily seen to be equivalent to:

$$\begin{bmatrix} \nabla X_{dst}(p) \\ \nabla Y_{dst}(p) \\ \nabla Z_{dst}(p) \end{bmatrix} = \begin{bmatrix} \nabla X_k(p) \\ \nabla Y_k(p) \\ \nabla Z_k(p) \end{bmatrix},\ \forall p \in \bar{\Psi}$$

We may then extract the destination vertices by solving 3 independent minimization problems (one for each of $X_{dst}, Y_{dst}, Z_{dst}$) which are all of the same type. It

therefore suffices to consider only one of them. E.g. for estimating Z_{dst} we need to find the solution to the following optimization problem:

$$\min_{Z_{dst}} \iint_{\bar{\Psi}} \|\nabla Z_{dst} - \nabla Z_k\|^2, \text{ with } Z_{dst}|_{\partial\bar{\Psi}} = \hat{Z}_{dst}|_{\partial\bar{\Psi}} \tag{5.5}$$

For discretizing the above problem we can make use of the underlying discrete pixel grid. To this end, we assume a 4-system neighborhood for the image pixels and we denote by $\mathcal{N}(p)$ the corresponding neighborhood of pixel p. In this case, the boundary $\partial\bar{\Psi}$ equals the set $\partial\bar{\Psi} = \{p \in \Psi : \mathcal{N}(p) \cap \bar{\Psi} \neq \emptyset\}$ and the finite-difference discretization of (5.5) yields the following quadratic optimization problem:

$$\min_{Z_{dst}} \sum_{p\in\bar{\Psi}} \sum_{q\in\mathcal{N}(p)} \big(Z_{dst}(p) - Z_{dst}(q) - [Z_k(p) - Z_k(q)]\big)^2 \text{ with } Z_{dst}(p) = \hat{Z}_{dst}(p),\ \forall p \in \partial\bar{\Psi} \tag{5.6}$$

This quadratic problem is, in turn, equivalent to the following system of linear equations:

$$|\mathcal{N}(p)| Z_{dst}(p) - \sum_{q\in\mathcal{N}(p)} Z_{dst}(q) = \sum_{q\in\mathcal{N}(p)} (Z_k(p) - Z_k(q)), \quad \forall p \in \bar{\Psi} \tag{5.7}$$

$$Z_{dst}(p) = \hat{Z}_{dst}(p), \quad \forall p \in \partial\bar{\Psi} \tag{5.8}$$

than can be solved with an iterative algorithm very efficiently due to the fact that all these linear equations form a sparse (banded) system.

Also, an alternative way of solving our optimization problem in (5.5) is by observing that any function minimizing (5.5) is also a solution to the following Poisson equation with Dirichlet boundary conditions [153]:

$$\triangle Z_{dst} = \mathbf{div}(\nabla Z_k), \text{ with } Z_{dst}|_{\partial\bar{\Psi}} = \hat{Z}_{dst}|_{\partial\bar{\Psi}} \tag{5.9}$$

Therefore, in this case, in order to extract the geometric maps $X_{dst}, Y_{dst}, Z_{dst}$ it suffices that we solve 3 independent Poisson equations of the above type. See Figures 5.11(c), 5.12(b) for a result produced with this method.

```
// Vertex shader
void main() {
 // set texture coordinates for multitexturing
 gl_TexCoord[0] = gl_MultiTexCoord0;
 gl_TexCoord[1] = gl_MultiTexCoord1;

 gl_Position = ftransform();
}

// Pixel shader
uniform float m; // the amount of morphing
uniform sampler2D tex0; //texture of image I_k
uniform smpaler2D tex1; //texture of image I_{k+1}

void main() {
 vec2 st0 = texture2D(tex0,gl_TexCoord[0].st);
 vec2 st1 = texture2D(tex1,gl_TexCoord[1].st);
 gl_FragColor = (1-m)*st0+m*st1;
}
```

Fig. 5.13: Pixel shader code (and the associated vertex shader code), written in GLSL (OpenGL Shading Language), for implementing the photometric morphing.

```
...

// enable vertex blending with 2 weights
glEnable(GL_VERTEX_BLEND_ARB);
glVertexBlendARB(2); ...

// set 1^st blending weight for MESH_Ψ^k, MESH_Ψ̄^k
glWeightfvARB( 1-m );

// you can now render MESH_Ψ^k, MESH_Ψ̄^k
glMatrixMode(GL_MODELVIEW0_ARB); ...

// set 2^nd blending weight for MESH_Ψ^dst, MESH_Ψ̄^dst
glWeightfvARB( m );

// you can now render MESH_Ψ^dst, MESH_Ψ̄^dst
glMatrixMode(GL_MODELVIEW1_ARB);
...
```

Fig. 5.14: Skeleton code in C for applying vertex blending in OpenGL.

5.7 Rendering pipeline

An important advantage of our framework is that, regardless of the scene's size, only one "morphable 3D-model" L_{morph} needs to be displayed at any time during rendering, i.e. the rendering pipeline has to execute the geometric and photometric morphing for only one local model L_k (as described in section 5.6). This makes our system extremely scalable to large scale scenes. In addition to that, by utilizing the enhanced capabilities of modern 3D graphics hardware, both types of morphing can admit a GPU[1] implementation, thus making our system ideal for 3D acceleration and capable of achieving very high frame rates during

[1]GPU stands for Graphics Processing Unit

rendering.

More specifically, for implementing the photometric morphing of model L_k, *multitexturing* needs to be employed as a first step. To this end, both images I_k, I_{k+1} will be used as textures and each 3D vertex whose corresponding 2D point $p \in I_k$ is located inside region Ψ will be assigned 2 pairs of texture coordinates: the first pair will coincide with the image coordinates of point $p \in I_k$, while the second one will be equal to the image coordinates of the corresponding point $u_{k \to k+1}(p) \in I_{k+1}$ (see (5.3)). Then, given these texture coordinates, a so-called *pixel-shader* (along with its associated *vertex-shader*) [118] can simply blend the two textures in order to implement (on the GPU) the photometric morphing defined by (5.2). Pixel and vertex shaders are user defined scripts that are executed by the GPU for each incoming 3D vertex and output pixel respectively. One possible implementation of such scripts, for the case of photometric morphing, is shown in Figure 5.13 where, for this specific example, the OpenGL Shading Language (GLSL) [118] has been used for describing the shaders. As for the 3D vertices, which are associated to points located inside region $\bar{\Psi}$, the situation is even simpler, since no actual photometric morphing takes place in there (see (5.4)) and so only image I_k needs to be texture-mapped onto these vertices.

On the other hand, for implementing the geometric morphing, the following procedure is used: two 2D triangulations of regions Ψ, $\bar{\Psi}$ are first generated resulting into two 2D triangle meshes $\text{TRI}_{\Psi}, \text{TRI}_{\bar{\Psi}}$. Based on these triangulations and the underlying geometric maps of L_k, two 3D triangle meshes $\text{MESH}^k_{\Psi}, \text{MESH}^k_{\bar{\Psi}}$ are constructed. Similarly, using $\text{TRI}_{\Psi}, \text{TRI}_{\bar{\Psi}}$ and the destination geometric maps $X_{dst}, Y_{dst}, Z_{dst}$, two more 3D triangle meshes $\text{MESH}^{dst}_{\Psi}, \text{MESH}^{dst}_{\bar{\Psi}}$ are constructed as well. It is then obvious that geometric morphing (as defined by (5.1)) amounts to a simple *vertex blending* operation i.e. meshes $\text{MESH}^k_{\Psi}, \text{MESH}^k_{\bar{\Psi}}$ are weighted by $1-m$, meshes $\text{MESH}^{dst}_{\Psi}, \text{MESH}^{dst}_{\bar{\Psi}}$ are weighted by m and the resulting weighted vertices are then added together. Vertex blending, however, is an operation that is directly supported by all modern GPUs and, as an example, Figure 5.14 contains skeleton code in C showing how one can implement vertex blending using the OpenGL standard.

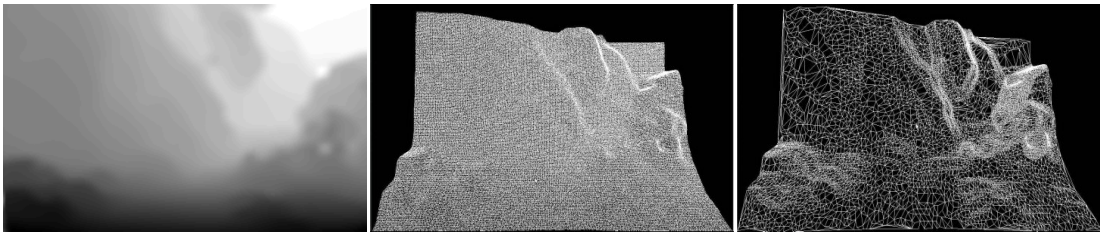
Therefore, based on the above observations, rendering a morphable model simply amounts to feeding into the GPU just 4 textured triangle meshes. This is,

however, a rendering path, which is highly optimized in all modern GPUs and, therefore, a considerable amount of 3D acceleration can be achieved this way during the rendering process.

5.7.1 Decimation of local 3D models

Up to now we have assumed that a full local 3D model is constructed each time, i.e. all points of the image grid are included as vertices in the 2D triangulations $\text{TRI}_{\Psi}, \text{TRI}_{\bar{\Psi}}$. However, we can also use simplified versions of these 2D triangle meshes, provided, of course, that these simplified meshes approximate well the underlying geometric maps. In fact, due to our framework's structure, a great amount of simplification can be achieved and the reason is that a simplified model L'_k has to be a good approximation to the full local model L_k only in the vicinity of pos_k (remember that model L_k is being used only in a local region around pos_k). Based on this observation, the following iterative procedure is being used for the simplification of the 2D meshes: at the start of each iteration there exists a current 2D Delaunay triangulation TRI_i, which has as vertices only a subset of the points on the image grid. Based on TRI_i, an error function $e(p)$ is defined over the image grid, which is measuring how well the current MESH_i approximates the underlying geometric maps (here MESH_i denotes the 3D surface defined by TRI_i). To each triangle, say T, of TRI_i we then associate the following two quantities: $e(T) = \max_{p \in T} e(p)$ (i.e. the maximum error across T) and $p(T) = \arg\max_{p \in T} e(p)$ (i.e. the interior point of T achieving this maximum error). At each iteration the triangle $T_{max} = \arg\max_{T \in \text{TRI}_i} e(T)$ of maximum error is selected and its point $p(T_{max})$ is added as a new vertex in the triangulation. This way a new Delaunay triangulation TRI_{i+1} is given as input to the next iteration of the algorithm and the process repeats until the maximum error $\max_{T \in \text{TRI}_i} e(T)$ falls below a user specified threshold e_{max}, which basically controls the total amount of simplification to be applied to the local model. Our algorithm is initialized with a sparse Delaunay triangulation TRI_0 and the only restriction imposed on TRI_0 is that it should contain the edges along the boundary between regions Ψ and $\bar{\Psi}$ (i.e. a constrained Delaunay triangulation has to be used) so that there are no cracks at the boundary of the corresponding meshes.

For completely specifying the decimation process, all that remains to be de-



(a) (b) (c)

Fig. 5.15: (a) Estimated disparity field corresponding to a local 3D model L_k. **(b)** Resulting full 3D model produced when a non-decimated 2D triangulation of the geometric maps has been used. **(c)** Simplified 3D model of L_k produced using a decimated 2D triangulation where the e_{max} threshold has been set equal to 0.5 pixels.

fined is the error function $e(p)$. One option would be to set $e(p) = ||\text{DEV}(p)||$, where $\text{DEV}(p) = ||\text{MESH}_i(p) - [X_k(p)\ Y_k(p)\ Z_k(p)]\,||$ denotes the geometric deviation at p between MESH_i and the underlying geometric maps ($\text{MESH}_i(p)$ is the 3D point defined by MESH_i at p). However, based on the fact that MESH_i needs to approximate L_k well only in a local region between positions pos_k and pos_{k+1} of the path, we choose to relate $e(p)$ to the maximum projection error at these locations. More specifically, we set:

$$e(p) = \max(\text{PROJ\_ERR}_{pos_k}, \text{PROJ\_ERR}_{pos_{k+1}}),$$

where PROJ_ERR_{pos_k} and $\text{PROJ\_ERR}_{pos_{k+1}}$ denote the maximum projection error at positions pos_k and pos_{k+1} respectively, i.e.:

$$\text{PROJ\_ERR}_{pos_k} = \max_{p \in I_k} ||\text{PROJ}_{pos_k}(\text{DEV}(p))||$$

$$\text{PROJ\_ERR}_{pos_{k+1}} = \max_{p \in I_{k+1}} ||\text{PROJ}_{pos_{k+1}}(\text{DEV}(p))||$$

In practice this definition of the error $e(p)$ has given excellent results and managed to achieve much larger reductions in the geometric complexity of the local 3D models. Furthermore, the user-defined threshold e_{max} can now be expressed in pixel units and can thus be set in a far more intuitive way by the user. An example of a simplified local model that has been produced in this manner is shown in Figure 5.15, in which case e_{max} has been set equal to 0.5 pixels. We should finally note that by using the simplified local 3D models one can reduce the rendering time even further, thus achieving higher frame rates e.g. over 60fps.

5.8 3D-mosaics construction

Up to this point we have been assuming that during the image acquisition process, we have been capturing one stereoscopic image-pair per key-position along the path. We will now consider the case in which multiple stereoscopic views per key-position are captured and these stereoscopic views are related to each other by a simple rotation of the stereoscopic camera. This scenario is very useful in cases where we need to have an extended field of view (like in large VR screens) and/or when we want to be able to look around the environment. In this new case, multiple local 3D models per key-position will exist and they will be related to each other by a pure rotation in 3D space.

In order to reduce this case to the one already examined, it suffices that a single local model per key-position (called *3D-mosaic* hereafter) is constructed. This 3D model should replace all local models at that position. Then at any time during the rendering process, a morphing between a successive pair of these new local models (3D-mosaics) needs to take place as before. For this reason, the term *"morphable 3D-mosaics"* is being used in this case.

As already explained, a 3D-mosaic at a certain position along the path should replace/comprise all local models coming from captured stereoscopic views at that position. Let $L_i = (X_i, Y_i, Z_i, I_i, dom_i)$ with $i \in \{1, \ldots, n\}$ be such a set of local models. Then a new local model $L_{mosaic} = (X_{mosaic}, Y_{mosaic}, Z_{mosaic}, I_{mosaic}, dom_{mosaic})$ needs to be constructed, which amounts to filling its geometric and photometric maps. Intuitively, L_{mosaic} should correspond to a local model produced from a stereoscopic camera with a wider field of view placed at the same path position.

It is safe to assume that the images I_i (which are the left-camera images), correspond to views related to each other by a pure rotation. (Actually, the relative pose between 2 such images will not be pure rotation but will also contain a small translational part due to the fact that the stereoscopic camera rotates around the tripod and not the optical center of the left camera. However this translation is negligible in practice). We may therefore assume that the local 3D models are related to each other by a pure rotation as well. An overview of the steps that needs to be taken for the construction of L_{mosaic} now follows:

- As a first step the rotation between local models needs to be estimated. This

will help us in registering the local models in 3D space.

- Then a geometric rectification of each L_i must take place so that the resulting local models are geometrically consistent with each other. This is a necessary step since the geometry of each L_i has been estimated only approximately and thus contains errors.

- Eventually, the maps of the refined and consistent local models will be merged so that the final map of the 3D-mosaic is produced

The most interesting problem that needs to be handled during the 3D-mosaic construction is that of making all models geometrically consistent so that a seamless (without discontinuities) geometric map of L_{mosaic} is produced. Each of the above steps will be explained in the following sections.

5.8.1 Rotation (R_{ij}) estimation between views I_i, I_j

First the homography H_{ij} between images I_i, I_j will be computed. (Since the views I_i, I_j are related by a rotation, H_{ij} will be the infinite homography induced by the plane at infinity.) For the H_{ij} estimation [60], a sparse set of (at least 4) point matches between I_i, I_j is first extracted and then a robust estimation procedure (e.g. RANSAC) is applied to cope with outliers. Inlier matches can then be used to refine the H_{ij} estimate by minimizing a suitable error function.

If $R_{ij} \in SO(3)$ is the 3×3 orthonormal matrix representing rotation, then: $H_{ij} = K_{left} R_{ij} K_{left}^{-1} \Leftrightarrow R_{ij} = K_{left}^{-1} H_{ij} K_{left}$. In practice due to errors in the computed H_{ij}, the above matrix will not be orthonormal. So for the estimation of R_{ij} [129], an iterative minimization procedure will be applied to: $\sum_k dist(p'_k, K_{left} R_{ij} K_{left}^{-1} p_k)^2$, where (p_k, p'_k) are the inlier matches that resulted after estimation of H_{ij} while $dist$ denotes euclidean image distance. The projection of $K_{left}^{-1} H_{ij} K_{left}$ to the space $SO(3)$ of 3D rotation matrices will be given as initial value to the iterative procedure.

5.8.2 Geometric rectification of local models

Since at this stage the rotation between any two local models is known, hereafter we may assume that the 3D vertices of all L_i are expressed in a common 3D

coordinate system. Unfortunately L_i are not geometrically consistent with each other, so the model resulting from combining these local models directly, would contain a lot of discontinuities at the boundary between any 2 neighboring L_i (see Fig. 5.16(c)). This is true because L_i have been created independently and their geometry has been estimated only approximately.

Let $\mathrm{RECTIFY}_{L_i}(L_j)$ denote an operator which takes as input 2 local models, L_i, L_j, and modifies the geometric-maps only of the model L_j so that they are consistent with the geometric-maps of the model L_i (the geometric maps of L_i do not change during RECTIFY). Assuming that such an operator exists, then ensuring consistency between all models can be achieved by merely applying $\mathrm{RECTIFY}_{L_i}(L_j)$ for all pairs L_i,L_j with $i < j$.

So it suffices that we define $\mathrm{RECTIFY}_{L_i}(L_j)$ for any 2 models, say L_i, L_j. Let X, Y, Z be the new rectified geometric-maps of L_j that we want to estimate so that they are geometrically consistent with those of L_i. Since we know homography H_{ij}, we may assume that image points of L_i have been aligned to the image plane of L_j. Let $\Psi = dom_i \cap dom_j$ be the overlap region of the 2 models. To be geometrically consistent, the new rectified maps of L_j should coincide with those of L_i at points inside Ψ:

$$\Big[X(p)\, Y(p)\, Z(p)\Big] = \Big[X_i(p)\, Y_i(p)\, Z_i(p)\Big]\,,\ \forall p \in \Psi \tag{5.10}$$

We still need to define the rectified maps on $\bar{\Psi} = dom_j \backslash \Psi$. On one hand, this must be done so that no discontinuity appears along $\partial\bar{\Psi}$ (and thus seamless rectified maps are produced). On the other hand, we must try to preserve the relative 3D structure of the existing geometric-maps (of L_j) in the interior of $\bar{\Psi}$. The last statement amounts to:

$$\begin{bmatrix} X(p)-X(q) \\ Y(p)-Y(q) \\ Z(p)-Z(q) \end{bmatrix} = \begin{bmatrix} X_j(p)-X_j(q) \\ Y_j(p)-Y_j(q) \\ Z_j(p)-Z_j(q) \end{bmatrix} \quad \forall p, q \in \bar{\Psi}$$

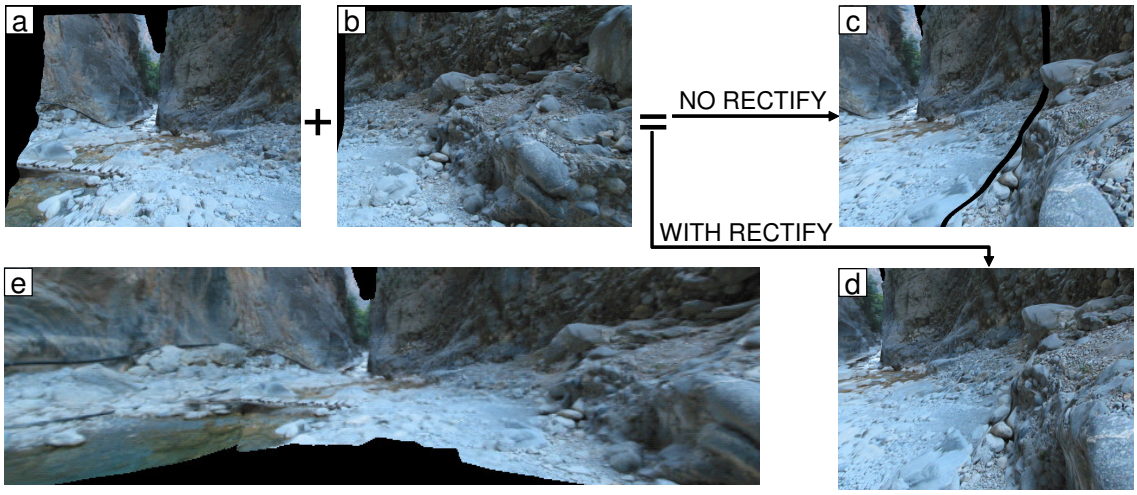


Fig. 5.16: Rendered views of: **(a)** a local model L_j **(b)** a local model L_i **(c)** a 3D-mosaic of L_i, L_j without geometry rectification (holes are due to errors in the geometry of the local models and not due to misregistration in 3D space) **(d)** a 3D-mosaic of L_i, L_j after $\text{RECTIFY}_{L_i}(L_j)$ has been applied **(e)** a bigger 3D-mosaic created from L_i, L_j as well as another local model which is not shown

or equivalently:

$$\begin{bmatrix} \nabla X(p) \\ \nabla Y(p) \\ \nabla Z(p) \end{bmatrix} = \begin{bmatrix} \nabla X_j(p) \\ \nabla Y_j(p) \\ \nabla Z_j(p) \end{bmatrix}, \ \forall p \in \bar{\Psi} \tag{5.11}$$

Then, based on (5.11) and (5.10), we can extract the Z rectified map (X, Y maps are treated similarly) by solving the following optimization problem:

$$\min_{Z} \iint_{\bar{\Psi}} \|\nabla Z - \nabla Z_j\|^2, \ Z|_{\partial\bar{\Psi}} = Z_i|_{\partial\bar{\Psi}} \tag{5.12}$$

The above problem, like the one defined by equation (5.5), can be reduced either to a banded linear system or to a Poisson differential equation, as explained in section 5.6.2. See Fig. 5.16(d) for a result produced with the latter method.

Another option for the merging of the geometric maps of L_i, L_j could have been the use of a feathering-like approach. The advantage of our approach (against feathering) is the preservation of the model's 3D structure. This can be illustrated with a very simple example (see Figure 5.17). Let a rectangular planar object be at constant depth Z_{true}. Suppose that depth map Z_i, corresponding to most of the left part of the object, has been estimated correctly ($Z_i \equiv Z_{true}$) but depth map Z_j, corresponding to most of the right part of the object, has been estimated as

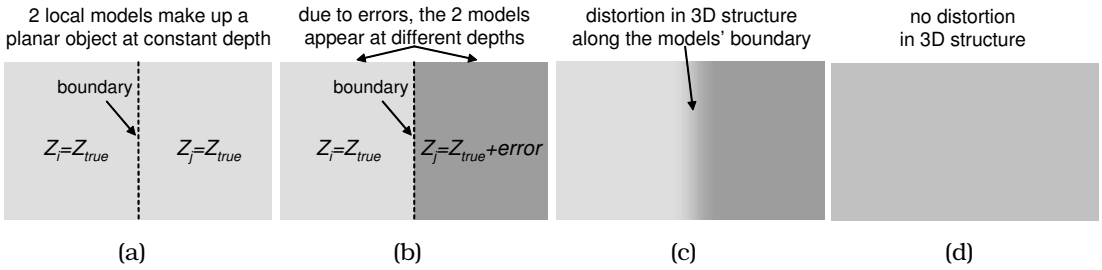


Fig. 5.17: A synthetic example illustrating the superiority of our approach against feathering (see also text). **(a)** True depth maps. **(b)** Estimated noisy depth maps. **(c)** Resulting 3D-mosaic's depth map using feathering. **(d)** Resulting 3D-mosaic's depth map using our method.

$Z_j \equiv Z_{true} + error$. When using a feathering-like approach, the resulting object will appear distorted in the center (its depth will vary from Z_{true} to $Z_{true} + error$ therein) and this distortion will be very annoying to the eye. On the contrary, by using our method, an object still having a planar structure will be produced. This is important since such errors often exist in models produced from disparity estimation. In fact, in this case, the errors' magnitude will be proportional to depth and can thus be quite large for distant (to the camera) objects like, e.g. local models of large scale scenes.

5.8.3 Merging the rectified local models

Since H_{ij} is known for any i, j, we may assume that all local models are defined on a common image plane. Therefore, due to the fact that the rectified geometric-maps are consistent with each other, we can directly merge them so that the $\{X, Y, Z\}_{mosaic}$ maps are produced. For the creation of the I_{mosaic} photometric map, a standard image-mosaicing procedure [129] can be applied independently. The valid region of the 3D-mosaic will be: $dom_{mosaic} = \cup_i dom_i$. Two 3D-mosaics that have been constructed in this manner appear in Figures 5.16(e) and 5.18.

5.9 Further results

As part of the DHX research project, the "morphable 3D-mosaic" framework has been already successfully applied to the visual 3D reconstruction of the well known Samaria Gorge in Crete (a gorge which is considered to be one of the most magnificent in the world and which was also awarded by the Council of Europe

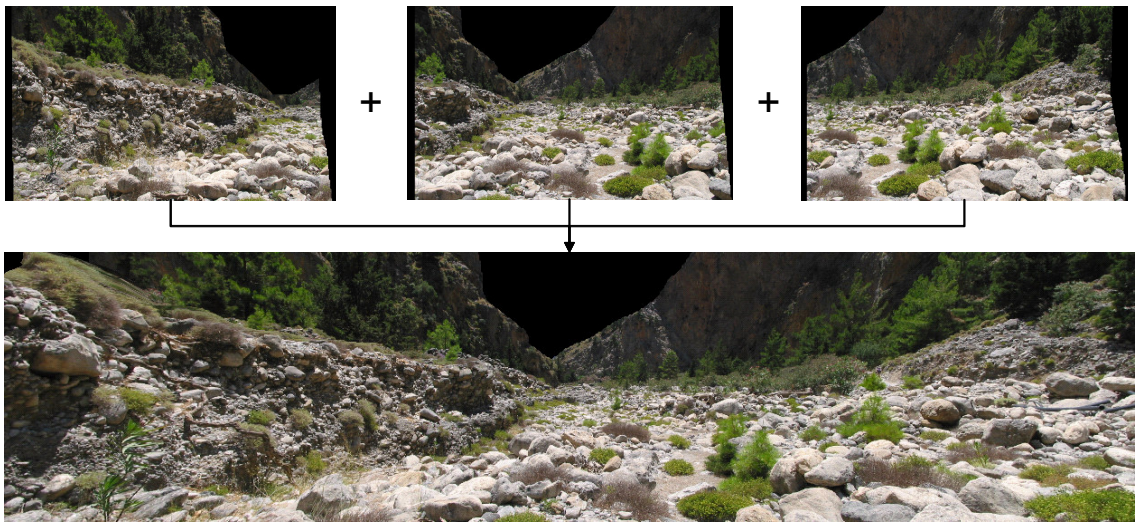
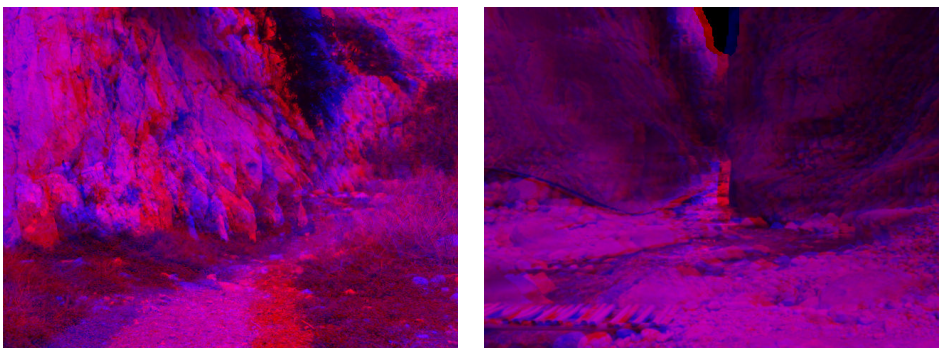


Fig. 5.18: Another example of a 3D-mosaic constructed using our method. **Top row:** three separate local 3D-models **Bottom row:** the resulting 3D-mosaic

with a Diploma First Class, as being one of Europe's most beautiful spots). Based on this 3D reconstruction, and by also using a 3D Virtual Reality installation, the ultimate goal of that work has been to provide a lifelike virtual tour of the Samaria Gorge to all visitors of the National History Museum of Crete, located in the city of Heraklion. To this end, the most beautiful spots along the gorge have been selected and for each such spot a predefined path, that was over 100 meters long, was chosen as well. About 15 key-positions have been selected along each path and approximately 45 stereoscopic views have been acquired at these positions with 3 stereoscopic views corresponding to each position (this way a 120^{o} wide field of view has been covered). Using the reconstructed "morphable 3D-mosaics", a photorealistic walkthrough of the Samaria Gorge has been obtained, which was visualized at interactive frame rates by means of a virtual reality system. The hardware equipment that has been used for the virtual reality system was a PC (with a Pentium 4 2,4GHz CPU on it) which was connected to a single-channel stereoscopic projection system from Barco consisting of a pair of circular polarized LCD projectors (Barco Gemini), an active-to-passive stereo converter as well as a projection screen. The rendering was done on a GeForce 6800 3D graphics card (installed on the PC) and, for the stereoscopic effect to take place, 2 views (corresponding to the left and right eye) were rendered by the graphics card at any time. Museum visitors were then able to participate in the virtual tour simply by wearing stereo glasses that were matched to the circular polarization of

the projectors. Two sample stereoscopic views, as would be rendered by the VR hardware, are shown in Figure 5.19. Despite the fact that a single graphics card has been used, very high frame rates of about 30fps in stereo mode (i.e. 60fps in mono mode) were obtained thanks to the optimized rendering pipeline provided by our framework. A sample from the obtained rendering results (that were generated in real time) are shown in Figure 5.20 for two different morphable models. In each row of that figure the leftmost and rightmost images represent rendered views of the model L_k and L_{k+1} respectively, while the images in between represent intermediate views of the morphable model along the path. Also, in Figure 5.21, we show some more rendered views where, this time, the virtual camera traverses a path containing more than one morphable 3D models.

Another difficulty that we had to face, during the visual reconstruction of the Samaria Gorge, was related to the fact that a small river was passing through a certain part of the gorge. This was a problem for the construction of the local 3D models as our stereo matching algorithm could not possibly extract disparity (i.e. find correspondences) for the points on the water surface. This was so because the water was moving and, even in places where it was static, sun reflections that existed on its surface were violating the lambertian assumption during stereo matching (see Figure 5.22(a)). Therefore, the disparity for all pixels lying on the water had to be estimated in a different way. To this end, as the water surface was approximately planar, a 2D homography (i.e. a 2D projective transformation represented by a 3×3 homogeneous matrix H_{water}), directly mapping left-image pixels on the water to their corresponding points in the right image, was estimated.



(a) (b)

Fig. 5.19: Two stereoscopic views as would be rendered by the VR system (for illustration purposes these are shown in the form of red-blue images).

(a) Sample views of a morphable 3D model. Each view corresponds to a different amount of morphing.

(b) Sample views (each with a different amount of morphing) for another morphable 3D model.

Fig. 5.20: Each row contains sample rendered views of a separate morphable 3D model. In each row the leftmost, rightmost images correspond to the views at pos_k, pos_{k+1} respectively.

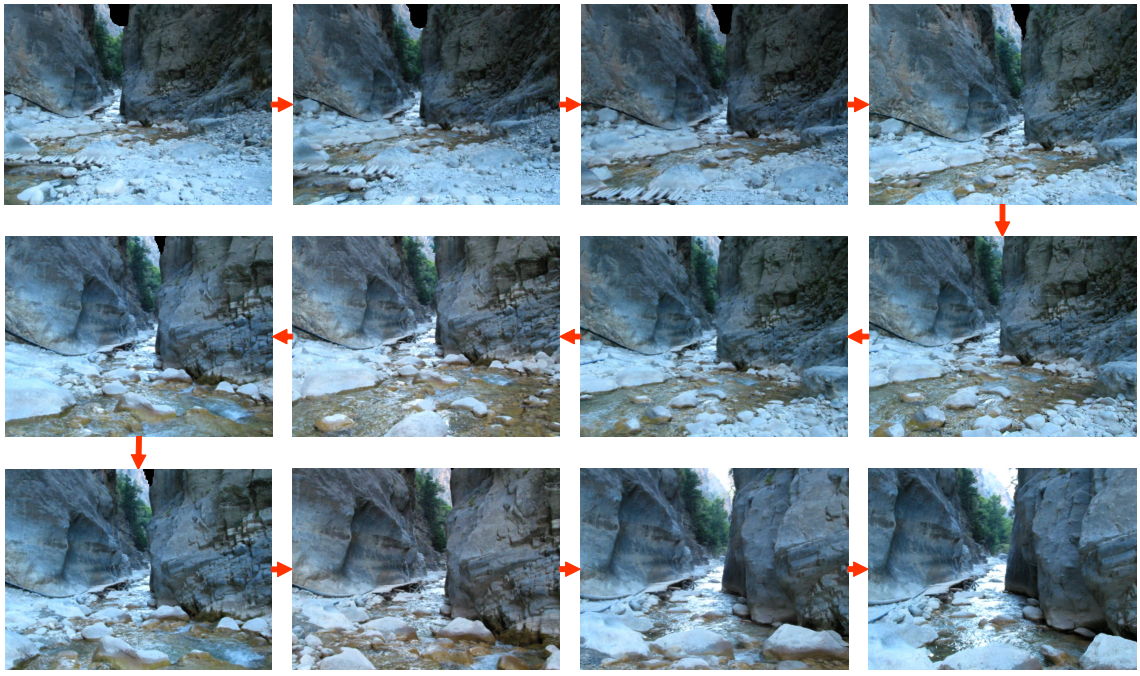


Fig. 5.21: Some rendered views that are produced as the virtual camera traverses a path through the so-called "Iron Gates" area, which is the most famous part of the Samaria Gorge. In this case the virtual camera passes through a series of successive morphable 3D models.

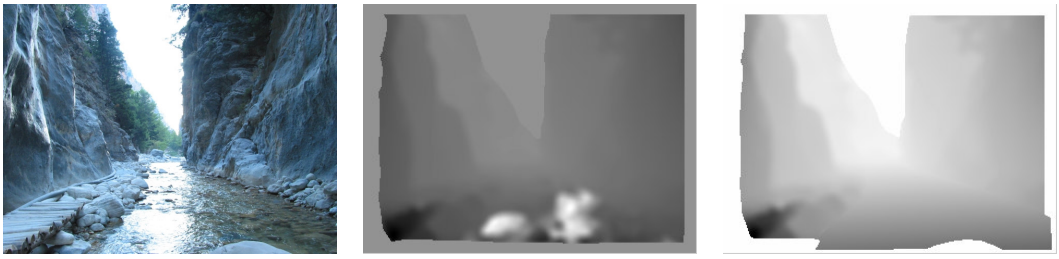
For estimating H_{water}, we made use of the fact that most left-image pixels that are located on the ground next to the river lie approximately at the same plane as the water surface and, in addition to that, stereo matching can extract valid correspondences for these pixels, as they are not on the water. A set $\{g_i\}_{i=1}^{K}$ of such

pixels in the left image is thus extracted and their matching points $\{g_i'\}_{i=1}^{K}$ in the right image are also computed based on the already estimated disparity maps. The elements of H_{water} can then be easily recovered by minimizing, through a robust procedure like RANSAC, the total reprojection error i.e. the sum of distances between $\{H_{water} \cdot g_i\}_{i=1}^{K}$ and $\{g_i'\}_{i=1}^{K}$. An example of a disparity field that has been estimated with this method can be seen in Figure 5.22(c). We should note that, by using a similar method, a 2D homography $H_{water}^{k \to k+1}$, mapping pixels of I_k lying on the water to their corresponding pixels in image I_{k+1}, can be computed as well. This way we can also manage to estimate optical flow $u_{k \to k+1}$ for all pixels of image I_k including those pixels of I_k that lie on the water.

Finally, we should mention that one of the additional benefits of having a virtual 3D reconstruction of the gorge is the ability e.g. to add synthetic visual effects or integrate synthetic objects into the environment. This way the visual experience of a virtual tour inside the gorge can be enhanced even further. For example, in Figure 5.23(a), we are showing some rendered views of the gorge, where we have also added synthetically generated volumetric fog, while, in Figures 5.23(b) and 5.23(c), we show a synthetic view where an *agrimi* (a wild goat which can be found only in the area of the Samaria Gorge), as well as an oleander plant has been integrated into the 3D virtual environment.

5.10 Conclusions

In conclusion, we have presented a new approach for obtaining photorealistic and interactive walkthroughs of large, outdoor scenes. To this end a new hybrid data structure has been presented, which is called “morphable 3D-mosaics”. No global model of the scene needs to be constructed and at any time during the rendering process, only one “morphable 3D-mosaic” is displayed. This enhances the scalability of the method to large environments. In addition, the proposed method uses a rendering path, which is highly optimized in modern 3D graphics hardware and thus can produce photorealistic renderings at interactive frame rates. In the future we intend to extend our rendering pipeline so that it can also take into account data from sparse stereoscopic views that have been captured at locations throughout the scene and not just along a predefined path. This could further



(a) (b) (c)

Fig. 5.22: (a) The left image of a stereoscopic image pair that has been captured at a region passing through a small river. **(b)** The estimated disparity by using a stereo matching procedure. As expected, the disparity field contains a lot of errors for many of the points on the water surface. This is true especially for those points that lie near the sun reflections on the water. **(c)** The corresponding disparity when a 2D homography is being used to fill the left-right correspondences for the points on the water. In this case the water surface is implicitly approximated by a 3D plane.

(a)

(b) (c)

Fig. 5.23: (a) Some rendered views of the gorge that also contain a synthetically generated volumetric fog. **(b)** A rendered view where a synthetic 3D model of the *agrimi*, a wild animal which is specific to the Samaria Gorge, has been integrated into the 3D virtual environment. **(c)** Another view with an oleander plant integrated as well.

enhance the quality of the rendered scene and would also permit a more extensive exploration of the virtual environment. Moreover, this extension still fits perfectly to the current architecture of the 3D-accelerated rendering pipeline (a blending of multiple local models will still be taking place). Furthermore, we intend to eliminate the need for a calibration of the stereoscopic camera, as well as to allow the stereo baseline to vary during the acquisition of the various stereoscopic views (this will make the data acquisition process even easier). Another issue that we

want to investigate is the ability of capturing the dynamic appearance of any time varying natural phenomena, such as moving water or grass that are frequently encountered in outdoor scenes (instead of just rendering these objects as static). To this end we plan to enhance our "morphable 3D-mosaic" framework so that it can also make use of real video textures that have been previously captured inside the scene. One limitation of our method is that it currently assumes that the lighting conditions across the scene are not drastically different (something which is not always true in outdoor environments). One approach, for dealing with this issue, is to obtain the radiometric response function of each photograph as well.

APPENDIX A

Technical proofs for theorems of chapter 3

A.1 Proof of theorem 3.2 about the optimality properties of the PD1 algorithm

The main purpose of this section is to provide a technical proof for theorem 3.2 of chapter 3, which basically states that the PD1 algorithm can always generate an f_{app}-approximate solution (in the meanwhile we will also provide a proof for Lemma 3.1). To this end, we will first start by stating some useful lemmas.

Lemma A.1. *Given any pair of primal-dual solutions (x, y) to the primal and dual LPs (Linear Programs) of Metric Labeling, then the following relationship between the value of the "APF" and the "loads" holds true:*

$$APF^{x,y} = \sum_{p} c_{p,x_p} + \sum_{(p,q)\in E} load_{pq}^{x,y} \tag{A.1}$$

Proof:

$$\begin{aligned} APF^{x,y} &= \sum_{p} ht_{p,x_p}^{y} = \sum_{p}\Big(c_{p,x_p} + \sum_{q:q\sim p} y_{pq,x_p}\Big) = \sum_{p} c_{p,x_p} + \sum_{(p,q)\in E}\Big(y_{pq,x_p} + y_{qp,x_q}\Big) \\ &= \sum_{p} c_{p,x_p} + \sum_{(p,q)\in E} load_{pq}^{x,y} \end{aligned}$$

□

Lemma A.2 (Flow conservation). *The components $\{f_p\}_{p\in V}$, $\{f_{pq}\}_{(p,q)\in E}$ of any valid flow passing though the capacitated graph $G_c^{x,y}$ satisfies the following equations:*

$$f_p = \sum_{q:q\sim p} \big(f_{pq} - f_{qp}\big) \tag{A.2}$$

Proof: It follows directly by applying the flow conservation at node p of the graph $G_c^{x,y}$. □

Lemma A.3. *Let p, q be two neighboring vertices i.e. $p \sim q$. Then, during an inner c-iteration of any of the algorithms $PD1, PD2_\mu, PD3$, the following properties hold true:*

(a) $a \neq c \Rightarrow \bar{y}_{pq,a}^{k+1} = \bar{y}_{pq,a}^{k},\ ht_{p,a}^{\bar{y}^{k+1}} = ht_{p,a}^{\bar{y}^{k}}$

(b) $x_p^k = c \Rightarrow x_p^{k+1} = c,\ \left(\bar{y}_{pq,c}^{k+1}, \bar{y}_{qp,c}^{k+1}\right) = \left(\bar{y}_{pq,c}^{k}, \bar{y}_{qp,c}^{k}\right),\ ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} = ht_{p,x_p^k}^{\bar{y}^{k}}$

(c) $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} \leq ht_{p,x_p^k}^{\bar{y}^{k}}$

(d) $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} \leq ht_{p,c}^{\bar{y}^{k+1}}$

(e) *if p is assigned label c, but q keeps its current label (i.e. $x_p^{k+1} = c$ and $x_q^{k+1} = x_q^k$), then $\bar{y}_{pq,c}^{k+1} = \bar{y}_{pq,c}^{k} + cap_{pq}$, i.e. the balance variable $\bar{y}_{pq,c}^{k+1}$ attains its maximum value*

(f) *(APF monotonicity) $APF^{x^{k+1},\bar{y}^{k+1}} \leq APF^{x^k,\bar{y}^k}$ Furthermore, if, at least one change of label has taken place during the current c-iteration, then $APF^{x^{k+1},\bar{y}^{k+1}} < APF^{x^k,\bar{y}^k}$*

Proof:

(a) This property follows directly from the fact that only the balance variables of the c labels are updated during a c-iteration, by definition.

(b) Due to $x_p^k = c$ and (3.18), the capacities of all interior edges $p\acute{q}, \acute{q}p$ with $\acute{q}$ adjacent to p (i.e. $\acute{q} \sim p$) will be zero and so no flow can pass through them i.e.:

$$f_{p\acute{q}} = f_{\acute{q}p} = 0 \quad \forall\ \acute{q} : \acute{q} \sim p \tag{A.3}$$

If we then apply the flow conservation at node p (A.2), we can see that the flow through edge sp will be zero as well (i.e. $f_p = 0$), which in turn implies that the edge sp is unsaturated (since $cap_{sp} = 1$ by (3.24)). Therefore by the reassigning rule it will also be $x_p^{k+1} = c$. Finally, the equality $\left(\bar{y}_{pq,c}^{k+1}, \bar{y}_{qp,c}^{k+1}\right) = \left(\bar{y}_{pq,c}^{k}, \bar{y}_{qp,c}^{k}\right)$ follows directly from applying (A.3) to $\acute{q} = q$ and then using (3.25), while the other equality $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} = ht_{p,x_p^k}^{\bar{y}^{k}}$ follows from $f_p = 0$ and (3.26).

(c) if $x_p^{k+1} \neq c$ then it will necessarily hold $x_p^{k+1} = x_p^k$, since by the reassign rule a vertex is either assigned label c or keeps its current label x_p^k. Therefore it will also be $x_p^k \neq c$. So by setting $x_p^{k+1} = x_p^k = a \neq c$ we may now apply property (a) and easily conclude that $ht^{\bar{y}^{k+1}}_{p,x_p^{k+1}} = ht^{\bar{y}^k}_{p,x_p^k}$, which means that the property holds in this case.

Therefore we may hereafter assume that $x_p^{k+1} = c$. In that case, if p is connected to the source node s then we may easily verify the property as follows:

$$\begin{aligned} ht^{\bar{y}^{k+1}}_{p,x_p^{k+1}} = ht^{\bar{y}^{k+1}}_{p,c} &= ht^{\bar{y}^k}_{p,c} + f_p && \text{by (3.26)} \\ &\leq ht^{\bar{y}^k}_{p,c} + cap_{sp} \\ &= ht^{\bar{y}^k}_{p,c} + \left(ht^{\bar{y}^k}_{p,x_p^k} - ht^{\bar{y}^k}_{p,c}\right) && \text{by (3.21)} \\ &= ht^{\bar{y}^k}_{p,x_p^k} \end{aligned}$$

Let us now consider the case where p is connected to the sink t: since we assume $x_p^{k+1} = c$ the reassign rule implies that there must be an unsaturated path, say $s \rightsquigarrow p$, from s to p. But it must then hold $f_p = cap_{pt}$ or else there will also be an unsaturated path $s \rightsquigarrow p \rightarrow t$ between the source and the sink which is impossible due to the max-flow min-cut theorem (see theorem 2.5 in chapter 2). Combining this fact (i.e. $f_p = cap_{pt}$) with (3.26) and the definition of cap_{pt} in (3.23) it then follows that:

$$\begin{aligned} ht^{\bar{y}^{k+1}}_{p,x_p^{k+1}} = ht^{\bar{y}^{k+1}}_{p,c} &= ht^{\bar{y}^k}_{p,c} - f_p \\ &= ht^{\bar{y}^k}_{p,c} - cap_{pt} \\ &= ht^{\bar{y}^k}_{p,c} - \left(ht^{\bar{y}^k}_{p,c} - ht^{\bar{y}^k}_{p,x_p^k}\right) = ht^{\bar{y}^k}_{p,x_p^k} \end{aligned}$$

(d) If $x_p^{k+1} = c$, the property obviously holds. So we may assume that $x_p^{k+1} \neq c$. In that case, it will also be $x_p^k \neq c$ as well (due to property (b)). If p is connected to the source node s, then the arc sp must be saturated, i.e. $f_p = cap_{sp}$ or else it would hold $x_p^{k+1} = c$ according to the reassign rule. Using this fact as well as (3.26) and the definition of cap_{sp} in (3.21) the

property then follows:

$$\begin{aligned} ht_{p,c}^{\bar{y}^{k+1}} &= ht_{p,c}^{\bar{y}^k} + f_p \\ &= ht_{p,c}^{\bar{y}^k} + cap_{sp} \\ &= ht_{p,c}^{\bar{y}^k} + \left(ht_{p,x_p^k}^{\bar{y}^k} - ht_{p,c}^{\bar{y}^k}\right) = ht_{p,x_p^k}^{\bar{y}^k} = ht_{p,x_p^{k+1}}^{\bar{y}^k} \end{aligned}$$

where the last equality is true due to the fact $x_p^k \neq c$ and property *(a)*.

On the other hand, if p is connected to the sink t then:

$$\begin{aligned} ht_{p,c}^{\bar{y}^{k+1}} &= ht_{p,c}^{\bar{y}^k} - f_p && \text{by (3.26)} \\ &\geq ht_{p,c}^{\bar{y}^k} - cap_{pt} \\ &= ht_{p,c}^{\bar{y}^k} - \left(ht_{p,c}^{\bar{y}^k} - ht_{p,x_p^k}^{\bar{y}^k}\right) && \text{by (3.23)} \\ &= ht_{p,x_p^k}^{\bar{y}^k} = ht_{p,x_p^{k+1}}^{\bar{y}^k} \end{aligned}$$

where again the last equality is true due to the fact $x_p^k \neq c$ and property *(a)*.

(e) If $x_q^k = c$ then $cap_{pq} = 0$ (due to (3.18)), while also $\bar{y}_{pq,c}^{k+1} = \bar{y}_{pq,c}^k$ (by property (b)) and so the property obviously holds. Therefore we may assume that $x_q^k \neq c$, which implies that $x_q^{k+1} \neq c$ as well (since $x_q^{k+1} = x_q^k$). Since p has been assigned the label c, there must exist an unsaturated path $s \rightsquigarrow p$ from s to p. But then the forward arc pq as well as the backward arc qp of the path $s \rightsquigarrow p \rightarrow q$ must be saturated i.e.:

$$f_{pq} = cap_{pq} \;\text{ and }\; f_{qp} = 0 \tag{A.4}$$

or else that path would also be unsaturated (which would in turn imply that $x_q^{k+1} = c$ contrary to our assumption above). Due to (A.4) and (3.25) the property then follows.

(f) The first inequality follows directly from *(c)* and the definition of the "APF". Furthermore, if at least one change of label has taken place then according to the reassign rule there must be at least one unsaturated arc, say sp, between the source and some node p. This implies that $f_p < cap_{sp}$ and so by also using (3.26) and the definition of cap_{sp} in (3.21) it is then trivial to show that $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} < ht_{p,x_p^k}^{\bar{y}^k}$. Due to this fact and by applying property (c) to

all other vertices the desired strict inequality follows.

□

Based on the previous lemma, we can directly prove lemma 3.1 of chapter 1 which is restated here for the reader's convenience.

Lemma A.4 (Corresponds to Lemma 3.1 of chapter 3). *Any pair of primal-dual solutions* $(x^{k+1}, \bar{y}^{k+1})$ *satisfies the following properties:*

Property 1: *If at least one vertex has changed its active label then, it holds true that:*

$$APF^{x^{k+1},\bar{y}^{k+1}} < APF^{x^k,\bar{y}^k}$$

Property 2: $ht^{\bar{y}^{k+1}}_{p,x_p^{k+1}} \leq ht^{\bar{y}^{k+1}}_{p,c}$

Property 3: *If* $c = x_p^{k+1} \neq x_q^{k+1}$, *then* $\bar{y}^{k+1}_{pq,c} = \bar{y}^k_{pq,c} + cap_{pq}$

Proof. Properties 1, 2 and 3 correspond directly to (f), (d), (e) of Lemma A.3 respectively. □

We can now turn to the proof of theorem 3.2 of chapter 3, which is the main theorem of this section. This theorem ensures the optimality properties of the PD1 algorithm and is restated here for convenience.

Theorem A.5 (Corresponds to theorem 3.2 of chapter 3). *The final primal and dual solutions generated by PD1 satisfy all conditions (3.13) - (3.16). Therefore, (as explained in section 3.4) these solutions are feasible and satisfy the relaxed complementary slackness conditions with* $f_1 = 1$, $f_2 = f_{app}$.

Proof: Due to the integrality assumption of the quantities $c_{p,a}, w_{pq}, d_{ab}$, both the initial dual solution as well as the capacities of the graph $G^{x^k,\bar{y}^k}$ are always of the form $\frac{n_0}{2}$ with $n_0 \in \mathbb{N}$. It can then be easily verified that any balance variable, and therefore the APF too, can take values only of that form. So after every c-iteration any decrease of APF will always have magnitude $\geq 1/2$. Based on this observation and the fact that, as mentioned above, POSTEDIT_DUALS does not alter the value of the APF function, the algorithm termination (i.e. no change of label taking place for $|L|$ consecutive inner iterations) is guaranteed by the APF monotonicity property A.3(f).

Feasibility conditions (3.13) are enforced by the definition of the PD1 algorithm (see Fig. 3.7). In addition, due to the specific assignment of capacities to interior edges (see (3.19)), the balance variables of edges pq, qp are not allowed to grow larger than $w_{pq}d_{min}/2$ and so constraints (3.16) are also enforced.

Furthermore, we can prove by induction that solutions x^k, y^k (for any k) satisfy slackness conditions (3.15) and have all of their active balance variables nonnegative i.e.

$$y^k_{pq,x^k_p} \geq 0 \tag{A.5}$$

These conditions are obviously true at initialization (by the definition of INIT_DUALS), so let us assume that they hold for x^k, y^k and let the current iteration be a c-iteration. We will then show that these conditions hold for x^{k+1}, y^{k+1} as well. To this end, we will consider 3 cases:

Case 1: let us first consider the case where $x^{k+1}_p = x^{k+1}_q = c$. Then $load^{x^{k+1},y^{k+1}}_{pq} = 0$ (due to (3.12)) and so (3.15) obviously holds, while (A.5) is guaranteed to be restored by the definition of POSTEDIT_DUALS.

Case 2: Next, let us examine the case where neither p nor q is assigned a new label (i.e. $x^{k+1}_p = x^k_p, x^{k+1}_q = x^k_q$). We can then show that:

$$y^{k+1}_{pq,x^{k+1}_p} = y^k_{pq,x^k_p} \quad y^{k+1}_{qp,x^{k+1}_q} = y^k_{qp,x^k_q} \tag{A.6}$$

and so both conditions (3.15), (A.5) follow directly from the induction hypothesis. Indeed, by applying either property A.3(a) or A.3(b), depending on whether $x^{k+1}_p = x^k_p = a \neq c$ or $x^{k+1}_p = x^k_p = c$, we conclude that $\bar{y}^{k+1}_{pq,x^{k+1}_p} = \bar{y}^k_{pq,x^k_p}$. In addition, $\bar{y}^k_{pq,x^k_p} = y^k_{pq,x^k_p} \geq 0$ with the equality being true due to the definition of the PREEDIT_DUALS function and the inequality following by the induction hypothesis. Combining the above relations we get:

$$\bar{y}^{k+1}_{pq,x^{k+1}_p} = y^k_{pq,x^k_p} \geq 0, \tag{A.7}$$

while with similar reasoning we can also show that:

$$\bar{y}^{k+1}_{qp,x^{k+1}_q} = y^k_{qp,x^k_q} \geq 0. \tag{A.8}$$

Therefore, both $\bar{y}^{k+1}_{pq,x^{k+1}_p}, \bar{y}^{k+1}_{qp,x^{k+1}_q}$ are nonnegative and so their values will not be

altered by POSTEDIT_DUALS:

$$y^{k+1}_{pq,x_p^{k+1}} = \bar{y}^{k+1}_{pq,x_p^{k+1}} \quad y^{k+1}_{qp,x_q^{k+1}} = \bar{y}^{k+1}_{qp,x_q^{k+1}} \tag{A.9}$$

The above equation, in conjunction with (A.7), (A.8), implies that (A.6) holds true, as claimed.

Case 3: Finally, let us consider the only remaining case according to which only one of p, q (say p) is assigned a new label c i.e. $x_p^{k+1} = c \neq x_p^k$, while the other one (say q) keeps its current label i.e. $x_q^{k+1} = x_q^k = a$ with $a \neq c$. In this case due to property A.3(e) and (3.19) it follows that $\bar{y}^{k+1}_{pq,c} = \bar{y}^k_{pq,c} + cap_{pq} = w_{pq} \cdot d_{min}/2$. In addition, it holds that $\bar{y}^{k+1}_{qp,a} = \bar{y}^k_{qp,a} = y^k_{qp,a} \geq 0$ where the 1^{st} equality is true due to $a \neq c$ and property A.3(a), the 2^{nd} equality is true due to the definition of PREEDIT_DUALS and the inequality follows from the induction hypothesis. Since $a \neq c$ (or equivalently $x_q^{k+1} \neq c$), POSTEDIT_DUALS (by definition) will alter none of the active balance variables $\bar{y}^{k+1}_{pq,c}, \bar{y}^{k+1}_{qp,a}$ and so $y^{k+1}_{pq,c} = \bar{y}^{k+1}_{pq,c}, y^{k+1}_{qp,a} = \bar{y}^{k+1}_{qp,a}$. By combining all of the above equalities it is now trivial to verify that (3.15), (A.5) hold for x^{k+1}, y^{k+1} as well.

Finally, to conclude the proof of this theorem we need to show that the last primal-dual pair of solutions satisfies condition (3.14). According to the termination criterion of the PD1 algorithm, during its last $|L|$ inner iterations there should be no label change. Let c be any label and consider the c-iteration out of these last $|L|$ iterations. During that iteration it will hold that:

$$ht^{\bar{y}^{k+1}}_{p,x_p^{k+1}} \leq ht^{\bar{y}^{k+1}}_{p,c}, \tag{A.10}$$

where the above inequality is true due to property A.3(d). In addition, since no change of label takes place, we can apply the same reasoning as in case 2 above and show again that (A.9) holds for any neighboring vertices p, q. This implies that all active balance variables are kept constant during the transition from $\bar{y}^{k+1}$ into y^{k+1} which in turn implies that $y^{k+1} = \bar{y}^{k+1}$, since, by definition, POSTEDIT_DUALS cannot touch any non-active balance variables. Therefore, the heights of labels do not change during the transition from $\bar{y}^{k+1}$ into y^{k+1} and so, based on the

previous inequality (A.10), it will also hold that:

$$ht^{y^{k+1}}_{p,x_p^{k+1}} \leq ht^{y^{k+1}}_{p,c} \tag{A.11}$$

Furthermore, the value of $ht^{y^{k+1}}_{p,x_p^{k+1}}$ is not altered during any of the next iterations. This is true because p keeps its current label (by the termination criterion) and so we may again show that (A.6) holds for all of the remaining iterations. Similarly, the value of $ht^{y^{k+1}}_{p,c}$ will not change hereafter, since by assumption this is the last c-iteration i.e. the last time the balance variables of the c labels are updated. Therefore, inequality (A.11) will be maintained until the end of the algorithm. Since the same reasoning can be applied to any label c, condition (3.14) will finally hold true at the end of the last iteration. □

A.2 Proof of theorem 3.3 about the optimality properties of the PD2$_\mu$ algorithm

The main result of this section will be to prove theorem 3.3 of chapter 3. That theorem guarantees that the PD2$_\mu$ can always generate a solution which is f_{app}-approximate in the worst case. Before that, however, we will need to state a few lemmas.

Lemma A.6. *During an inner c-iteration of the PD2$_\mu$ algorithm, let p, q be two neighbors (i.e. $p \sim q$) with $x_p^k = a, x_q^k = b$ and assume that $x^k, \bar{y}^k$ satisfy condition (3.27) i.e. $load_{pq}^{x^k,\bar{y}^k} = \mu w_{pq} d_{x_p^k x_q^k}$. Then the following properties hold true:*

(a) *if $a \neq c, b \neq c$, then $\bar{y}_{pq,c}^{k+1} \leq \mu w_{pq} d_{cb} - \bar{y}_{qp,b}^k$ and $\bar{y}_{qp,c}^{k+1} \leq \mu w_{pq} d_{ac} - \bar{y}_{pq,a}^k$*

(b) *$x^{k+1}, \bar{y}^{k+1}$ satisfy condition (3.27) as well, i.e. $load_{pq}^{x^{k+1},\bar{y}^{k+1}} = \mu w_{pq} d_{x_p^{k+1} x_q^{k+1}}$*

Proof:

(a) Since both of a, b are $\neq c$, then (3.31), (3.32), (3.33) hold true. In addition, by the lemma hypothesis:

$$load_{pq}^{x^k,\bar{y}^k} = \bar{y}_{pq,a}^k + \bar{y}_{qp,b}^k = \mu w_{pq} d_{ab} \tag{A.12}$$

By property A.3(e) the maximum value of $\bar{y}_{pq,c}^{k+1}$ will be $\bar{y}_{pq,c}^{k} + cap_{pq} = \bar{y}_{pq,c}^{k} + \mu w_{pq}(d_{ac} + d_{cb} - d_{ab}) = \mu w_{pq} d_{cb} - \bar{y}_{qp,b}^{k}$ where the first equality is due to (3.31) and the last equality follows by substituting d_{ab}, d_{ac} from (A.12), (3.33). Likewise the maximum value of $\bar{y}_{qp,c}^{k+1}$ will be $\bar{y}_{qp,c}^{k} + cap_{qp} = \bar{y}_{qp,c}^{k} = \mu w_{pq} d_{ac} - \bar{y}_{pq,a}^{k}$ where the first equality is due to (3.32) and the last equality follows from (3.33).

(b) If $x_p^{k+1} = x_q^{k+1}$, then $load_{pq}^{x^{k+1},\bar{y}^{k+1}} = 0$ and part (b) of the lemma obviously holds. Therefore we may hereafter assume that $x_p^{k+1} \neq x_q^{k+1}$. This assumption has as a result that not both of a, b can be equal to c or else it would hold $x_p^{k+1} = x_q^{k+1} = c$ due to property A.3(b). On the other hand, if either one of a, b is equal to c (say $x_p^k = a = c, x_q^k = b \neq c$), this implies that $\bar{y}_{qp,b}^{k+1} = \bar{y}_{qp,b}^{k}$ (due to $b \neq c$ and property A.3(a)) as well as $x_p^{k+1} = c$ and $\bar{y}_{pq,c}^{k+1} = \bar{y}_{pq,c}^{k}$ (due to $x_k^p = c$ and property A.3(b)). Then necessarily $x_q^{k+1} = x_q^k = b$ (since we assume $x_q^{k+1} \neq x_p^{k+1} = c$ and by definition of x^{k+1} any vertex q is either assigned label c or else keeps its current label x_q^k). By combining all of the above equalities it follows that $load_{pq}^{x^{k+1},\bar{y}^{k+1}} = \bar{y}_{pq,c}^{k+1} + \bar{y}_{qp,b}^{k+1} = \bar{y}_{pq,c}^{k} + \bar{y}_{qp,b}^{k} = load_{pq}^{x^k,\bar{y}^k} = \mu w_{pq} d_{cb}$ (where the last equality is true due to the lemma hypothesis) and part (b) of the lemma therefore holds in this case.

We still need to consider only the case where both of a, b are different than c (i.e. $a \neq c, b \neq c$). Since we assume $x_p^{k+1} \neq x_q^{k+1}$ only one of p, q may be assigned label c by x^{k+1}. If label c is assigned to p but q keeps its current label b (i.e. $x_p^{k+1} = c, x_q^{k+1} = b$), then by property A.3(e) $\bar{y}_{pq,c}^{k+1}$ attains its maximum value and so by part *(a)* $\bar{y}_{pq,c}^{k+1} = \mu w_{pq} d_{cb} - \bar{y}_{qp,b}^{k}$. In addition $\bar{y}_{qp,b}^{k+1} = \bar{y}_{qp,b}^{k}$ (due to $b \neq c$ and property A.3(a)) and so $load_{pq}^{x^{k+1},\bar{y}^{k+1}} = \bar{y}_{pq,c}^{k+1} + \bar{y}_{qp,b}^{k+1} = \left(\mu w_{pq} d_{cb} - \bar{y}_{qp,b}^{k}\right) + \bar{y}_{qp,b}^{k} = \mu w_{pq} d_{cb}$. Likewise we can show that if label c is assigned to q (by x^{k+1}) but p keeps its current label a, then $load_{pq}^{x^{k+1},\bar{y}^{k+1}} = \mu w_{pq} d_{ac}$. □

Lemma A.7. *During the kth inner iteration of the PD2$_\mu$ algorithm, the primal-dual solutions x^{k+1}, y^{k+1} (resulting after applying the* POSTEDIT_DUALS *routine) satisfy the following properties:*

(a) $load_{pq}^{x^{k+1},y^{k+1}} = load_{pq}^{x^{k+1},\bar{y}^{k+1}}$

(b) *The primal-dual solutions x^{k+1}, y^{k+1} satisfy conditions (3.27) i.e.* $load_{pq}^{x^{k+1},y^{k+1}} = \mu w_{pq} d_{x_p^{k+1} x_q^{k+1}}$

(c) $APF^{x^{k+1},y^{k+1}} = APF^{x^{k+1},\bar{y}^{k+1}}$

(d) $y^{k+1}_{pq,a} \leq |\bar{y}^{k+1}_{pq,a}|, \;\; y^{k+1}_{qp,a} \leq |\bar{y}^{k+1}_{qp,a}| \;\; \forall \, a \in L$

(e) $y^{k+1}_{pq,x^{k+1}_p} \geq 0, \;\; y^{k+1}_{qp,x^{k+1}_q} \geq 0$

Proof: As already mention in chapter 3, the role of POSTEDIT_DUALS is to edit dual solution $\bar{y}^{k+1}$ into y^{k+1} so that all active balance variables of y^{k+1} become nonnegative. To this end, POSTEDIT_DUALS is applying an operator RECTIFY(p,q) to any pair $(p,q) \in E$.

This operator is defined as follows: let $x^{k+1}_p = a, x^{k+1}_q = b$ and let us also assume that at least one of the active balance variables $\bar{y}^{k+1}_{pq,a}, \bar{y}^{k+1}_{qp,b}$ is negative, say $\bar{y}^{k+1}_{qp,b} < 0$. Then if $a = b$ the operator RECTIFY(p,q) simply sets $y^{k+1}_{pq,a} = y^{k+1}_{qp,a} = 0$, while if $a \neq b$ it sets $y^{k+1}_{pq,a} = \bar{y}^{k+1}_{pq,a} + \bar{y}^{k+1}_{qp,b}, \; y^{k+1}_{qp,b} = 0$.[1] During the transition from $\bar{y}^{k+1}$ to y^{k+1} no other balance variables are modified by the RECTIFY operator. Based on this definition of the RECTIFY operator, we can now start the proof of the current lemma.

(a) This equality can be easily verified directly from the definition of the operator RECTIFY.

(b) This property follows easily by induction from lemma A.6(b). Indeed, assuming that the pair x^k, y^k satisfies conditions (3.27), then the same thing applies to pair $x^k, \bar{y}^k$ as well due to (3.34). Therefore the hypothesis of lemma A.6 holds and by use of A.6(b) in that lemma the pair $x^{k+1}, \bar{y}^{k+1}$ also satisfies (3.27). By then applying property (a) the same conclusion can be drawn regarding the pair x^{k+1}, y^{k+1} and this completes the induction.

(c) It follows by combining property (a) above and equation (A.1).

(d) This can be trivially verified based on the definition of operator RECTIFY(p,q).

(e) Combining properties (a) and (b) we conclude that $load^{x^{k+1},\bar{y}^{k+1}}_{pq} \geq 0$. Using this fact it is then trivial to verify the property based on the definition of the RECTIFY operator.

□

[1] Of course we also set their conjugate balance variables as: $y^{k+1}_{qp,a} = -y^{k+1}_{pq,a}, \;\; y^{k+1}_{pq,b} = -y^{k+1}_{qp,b}$.

We are now ready to fulfill the main goal of this section, i.e. to provide a proof for theorem 3.3 of chapter 3 which basically ensures that the solutions generated by PD2$_\mu$ are always close to the optimal solutions. Along with its proof, we also repeat that theorem here for the reader's convenience.

Theorem A.8 (Corresponds to theorem 3.3 of chapter 3). *The final primal-dual solutions generated by PD2$_\mu$ are feasible and satisfy the relaxed complementary slackness conditions with $f_1 = \mu f_{app}$ and $f_2 = f_{app}$.*

Proof: Due to the integrality assumption of the quantities $c_{p,a}, w_{pq}, d_{ab}$, both the initial dual solution as well as the capacities of the graph $G^{x^k,\bar{y}^k}$ are always of the form $\frac{n_0}{2}$ with $n_0 \in \mathbb{N}$. It is then easy to verify that the APF function can take values only of the form $\frac{n_0}{2}$ with $n_0 \in \mathbb{N}$, so any decrease of APF will necessarily be of magnitude $\geq \frac{1}{2}$. The algorithm termination is then guaranteed by the APF monotonicity property A.3(f) and the observation that neither PREEDIT_DUALS (due to (3.35)) nor POSTEDIT_DUALS (due to property A.7(c)) alters the value of APF.

Also, conditions (3.28) are enforced by the definition of the $PD2_\mu$ algorithm (see Fig. 3.8), while conditions (3.27) follows directly from property A.7(b).

In order to prove that conditions (3.30) hold as well it is enough to show by induction that $y^k_{pq,c}, y^k_{qp,c} \leq \mu w_{pq} d_{max}\ \forall c \in L$. This is obviously true at initialization so let's assume it holds for $y^k_{pq,c}, y^k_{qp,c}$. We will show that during a c-iteration this holds for $y^{k+1}_{pq,c}, y^{k+1}_{qp,c}$ as well. Due to property A.7(d) it is enough to show $\bar{y}^{k+1}_{pq,c}, \bar{y}^{k+1}_{qp,c} \leq \mu w_{pq} d_{max}$. If either one of $x^k_p = a, x^k_q = b$ equals c, then by property A.3(b) $\bar{y}^{k+1}_{pq,c} = \bar{y}^k_{pq,c},\ \bar{y}^{k+1}_{qp,c} = \bar{y}^k_{qp,c}$. Also, $\bar{y}^k_{pq,c} = y^k_{pq,c},\ \bar{y}^k_{qp,c} = y^k_{qp,c}$ (since PREEDIT_DUALS may alter $y^k_{pq,c}$ only if $x^k_p \neq c$ and $x^k_q \neq c$). The assertion then follows from the induction hypothesis.

If both of a, b are $\neq c$ then by lemma A.6(a) $\bar{y}^{k+1}_{pq,c} \leq \mu w_{pq} d_{cb} - \bar{y}^k_{qp,b}$, while also $\bar{y}^k_{qp,b} = y^k_{qp,b}$ (since $b \neq c$ and PREEDIT_DUALS, by definition, may alter only balance variables of the form $y^k_{pq,c}$ during a c-iteration). But $y^k_{qp,b} \geq 0$ since $x^k_q = b$ i.e. this is an active balance variable (see property A.7(e)). Therefore $\bar{y}^{k+1}_{pq,c} \leq \mu w_{pq} d_{cb} \leq \mu w_{pq} d_{max}$. Likewise, using again lemma A.6(a), we can prove that $\bar{y}^{k+1}_{qp,c} \leq \mu w_{pq} d_{ac} \leq \mu w_{pq} d_{max}$ and the assertion follows.

Finally, we may show that the last primal-dual pair of solutions (say x, y) also satisfies conditions (3.29), by following the same reasoning that has been used in the proof of theorem 3.2 to show the satisfiability of the equivalent conditions

(3.14). Therefore all conditions (3.27)-(3.30) hold true and so (as explained in section 3.5) the pair (x, y^{fit}) generated by DUAL_FIT will be feasible and will also satisfy all required slackness conditions, thus concluding the proof of the theorem.

□

A.3 Proving the equivalence between algorithm PD2$_{\mu=1}$ and the α-expansion min-cut algorithm

The main goal of this section will be to show that the algorithms PD2$_{\mu=1}$ and α-expansion are equivalent to each other. To this end, it suffices to prove that theorem 3.4 of chapter 3 holds true. The following lemmas are going to be needed for that purpose.

Lemma A.9. *Let x be a label assignment and y a dual solution (not necessarily feasible) satisfying the following conditions:*

$$load_{pq}^{x,y} \leq w_{pq} d_{x_p x_q} \; \forall (p,q) \in E \tag{A.13}$$

Let us also denote by $PRIMAL^x$ the value of the primal objective function at x. Under these assumptions it is always true that $APF^{x,y} \leq PRIMAL^x$, while if, in addition, conditions (A.13) *hold as equalities, then $APF^{x,y} = PRIMAL^x$.*

Proof: Equation (A.1) and the assumptions of the lemma imply:

$$\begin{aligned} APF^{x,y} &= \sum_p c_{p,x_p} + \sum_{(p,q)\in E} load_{pq}^{x,y} \\ &\leq \sum_p c_{p,x_p} + \sum_{(p,q)\in E} w_{pq} d_{x_p x_q} = PRIMAL^x \end{aligned}$$

□

Lemma A.10. *Let x^k, y^k be a primal dual pair of solutions at the start of a c-iteration of the PD2$_{\mu=1}$ algorithm. Let x' be any label assignment due to a c-expansion of x^k.*

(a) $APF^{x^{k+1},\bar{y}^{k+1}} \leq APF^{x',\bar{y}^{k+1}}$

(b) $APF^{x',\bar{y}^{k+1}} \leq PRIMAL^{x'}$

Proof:

(a) Since x' is a label assignment due to a c-expansion, this means that x' may either keep the current label x_p^k of a vertex p or assign label c to it. If $x'_p = x_p^k \neq c$ (i.e. x' keeps the current label of p) then by property A.3(c) $ht_{p,x_p^k}^{\bar{y}^{k+1}} \leq ht_{p,x_p^k}^{\bar{y}^k} = ht_{p,x'_p}^{\bar{y}^k} = ht_{p,x'_p}^{\bar{y}^{k+1}}$, where the last equality is true due to $x'_p \neq c$ and property A.3(a). On the other hand if $x'_p = c$ then by property A.3(d) $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} \leq ht_{p,c}^{\bar{y}^{k+1}} = ht_{p,x'_p}^{\bar{y}^{k+1}}$. So in any case $ht_{p,x_p^{k+1}}^{\bar{y}^{k+1}} \leq ht_{p,x'_p}^{\bar{y}^{k+1}}$ and therefore $APF^{x^{k+1},\bar{y}^{k+1}} \leq APF^{x',\bar{y}^{k+1}}$.

(b) Let us first recall that the following equation holds:

$$load_{pq}^{x^k,\bar{y}^k} = load_{pq}^{x^k,y^k} = w_{pq} d_{x_p^k x_q^k} \tag{A.14}$$

where the 1^{st} equality is due to the preservation of the load by PREEDIT_DUALS (see (3.34)) while the 2^{nd} one follows from the fact that all x^k, y^k generated by PD2$_{\mu=1}$ satisfy slackness conditions (3.27).

According to lemma A.9, to prove the assertion it is enough to show that $x', \bar{y}^{k+1}$ satisfy (A.13). If $x'_p = x'_q$ this is obviously true, since in that case $load_{pq}^{x',\bar{y}^{k+1}} = 0$ due to (3.12). If $x'_p = x_p^k, x'_q = x_q^k$ then by applying either property A.3(a) or A.3(b) (depending on whether $x_p^k = a \neq c$ or $x_p^k = c$) we can conclude that $\bar{y}_{pq,x_p^k}^{k+1} = \bar{y}_{pq,x_p^k}^k$. Similarly we can show that $\bar{y}_{qp,x_q^k}^{k+1} = \bar{y}_{qp,x_q^k}^k$ and so:

$$load_{pq}^{x^k,\bar{y}^{k+1}} = load_{pq}^{x^k,\bar{y}^k} \tag{A.15}$$

The property then follows since: $load_{pq}^{x',\bar{y}^{k+1}} = load_{pq}^{x^k,\bar{y}^{k+1}} = load_{pq}^{x^k,\bar{y}^k} = w_{pq} d_{x_p^k x_q^k} = w_{pq} d_{x'_p x'_q}$, where the first and last equalities are true due to our assumption that $x'_p = x_p^k, x'_q = x_q^k$, while the 2^{nd} and 3^{rd} equalities are true due to equations (A.15) and (A.14) respectively. Also, if $x'_p \neq c, x'_q \neq c$ then necessarily $x'_p = x_p^k, x'_q = x_q^k$ (since x' is a c-expansion) and so we fall back into the previous case.

Therefore we still need to consider only the case where $x'_p \neq x'_q$, $(x'_p, x'_q) \neq (x_p^k, x_q^k)$ and one of x'_p, x'_q is equal to c. Assume that $x'_p = c$ and let us also set $x_p^k = a, x_q^k = b$. Based on all of the above assumptions and the fact that

x' is a c-expansion of x^k, one may then easily prove that: $x'_q = b, a \neq c, b \neq c$. This together with (A.14) implies that the hypothesis of lemma A.6(a) holds and so $\bar{y}^{k+1}_{pq,c} \leq w_{pq}d_{cb} - \bar{y}^{k}_{qp,b}$ while also $\bar{y}^{k+1}_{qp,b} = \bar{y}^{k}_{qp,b}$ (due to $b \neq c$ and property A.3(a)). Therefore $load^{x',\bar{y}^{k+1}}_{pq} = \bar{y}^{k+1}_{pq,c} + \bar{y}^{k+1}_{qp,b} \leq \left(w_{pq}d_{cb} - \bar{y}^{k}_{qp,b}\right) + \bar{y}^{k}_{qp,b} = w_{pq}d_{cb}$ and the lemma follows. □

Based on the previous lemmas, we are now able to provide a proof for theorem 3.4, thus showing the equivalence between the α-expansion algorithm and our primal-dual algorithm PD2$_{\mu=1}$. Theorem 3.4 is restated next for the reader's convenience.

Theorem A.11 (Corresponds to theorem 3.4 of chapter3). *The label assignment x^{k+1} selected during a c-iteration of the PD2$_{\mu=1}$ algorithm, has the minimum primal cost among all label assignments that can result after a c-expansion of x^k.*

Proof: Assignment x^{k+1} is indeed a c-expansion, since no c label may be replaced during a c-iteration (see property A.3(b)). In addition $load^{x^{k+1},\bar{y}^{k+1}}_{pq} = load^{x^{k+1},y^{k+1}}_{pq} = w_{pq}d_{x^{k+1}_p x^{k+1}_q}$ due to properties A.7(a) and A.7(b). So, solutions $x^{k+1}, \bar{y}^{k+1}$ satisfy conditions (A.13) of lemma A.9 as equalities and therefore $PRIMAL^{x^{k+1}} = APF^{x^{k+1},\bar{y}^{k+1}}$ by that lemma. Let now x' be any other label assignment due to a c-expansion of x^k. Combining the above equality with the (a) and (b) inequalities of lemma A.10 we get: $PRIMAL^{x^{k+1}} = APF^{x^{k+1},\bar{y}^{k+1}} \leq APF^{x',\bar{y}^{k+1}} \leq PRIMAL^{x'}$ and the theorem therefore follows. □

Bibliography

[1] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision", In *Computational Models of Visual Processing*, MIT Press, Cambridge, MA, 3–20, 1991.

[2] Shivani Agarwal and Dan Roth, "Learning a sparse representation for object detection", In the proceedings of the *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, Springer-Verlag, London, UK, 113–130, 2002.

[3] Daniel G. Aliaga, Thomas Funkhouser, Dimah Yanovsky, and Ingrid Carlbom, "Sea of images", In the proceedings of the *VIS 2002*, 331–338, 2002.

[4] Amir A. Amini, Terry E. Weymouth, and Ramesh Jain, "Using dynamic programming for solving variational problems in vision." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, No. 9, 855–867, 1990.

[5] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talvar, and E. Tardos, "Approximate classification via earthmover metrics", In the proceedings of the *Proceedings of the 15^{th} Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004.

[6] Michael Ashikhmin, "Synthesizing natural textures", In the proceedings of the *Symposium on Interactive 3D Graphics*, 217–226, 2001.

[7] Coloma Ballester, Marcelo Bertalmío, Vicent Caselles, Guillermo Sapiro, and Joan Verdera, "Filling-in by joint interpolation of vector fields and gray levels." *IEEE Transactions on Image Processing*, vol. 10, No. 8, 1200–1211, 2001.

[8] Adrian Barbu and Song Chun Zhu, "Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, No. 8, 1239–1253, 2005.

[9] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques", In the proceedings of the *CVPR*, 236–242.

[10] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, 1962.

[11] Marcelo Bertalmío, "Contrast invariant inpainting with a 3rd order, optimal pde." In the proceedings of the *IEEE International Conference on Image Processing*.

[12] Marcelo Bertalmío, A. L. Bertozzi, and Guillermo Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting." In the proceedings of the *CVPR (1)*, 355–362, 2001.

[13] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester, "Image inpainting", In the proceedings of the *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 417–424, 2000.

[14] Marcelo Bertalmío, Luminita A. Vese, Guillermo Sapiro, and Stanley Osher, "Simultaneous structure and texture image inpainting." In the proceedings of the *CVPR (2)*, 707–712, 2003.

[15] J. Besag, "Spatial interaction and the statistical analysis of lattice systems", *Journal of the Royal Statistical Society, series B*, vol. 36, No. 2, 192–236, 1974.

[16] J. Besag, "On the statistical analysis of dirty pictures", *Journal of the Royal Statistical Society, series B*, vol. 48, 259–302, 1986.

[17] Kiran Bhat, Steven Seitz, Jessica K Hodgins, and Pradeep Khosla, "Flow-based video synthesis and editing", *ACM Transactions on Graphics (SIGGRAPH 2004)*, vol. 23, No. 3, August 2004.

[18] Stan Birchfield and Carlo Tomasi, "Depth discontinuities by pixel-to-pixel stereo", In the proceedings of the *ICCV*, 1073–1080, 1998.

[19] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields", *CVIU*, vol. 63, No. 1, 75–104, 1996.

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

[21] Oren Boiman and Michal Irani, "Detecting irregularities in images and in video", In the proceedings of the *International Conference On Computer Vision*, 2005.

[22] Jeremy S. De Bonet, "Multiresolution sampling procedure for analysis and synthesis of texture images", *Computer Graphics*, vol. 31, No. Annual Conference Series, 361–368, 1997.

[23] Jean Yves Bouguet, "Camera Calibration Toolbox for MATLAB", http://www.vision.caltech.edu/bouguetj/calib_doc.

[24] Yuri Boykov and Marie Pierre Jolly, "Interactive organ segmentation using graph cuts", In the proceedings of the *MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer-Verlag, London, UK, 276–286, 2000.

[25] Yuri Boykov and Marie Pierre Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images." In the proceedings of the *IEEE International Conference on Computer Vision*, 105–112, 2001.

[26] Y. Boykov, O. Veksler, and R. Zabih, "Markov random fields with efficient approximations", In the proceedings of the *IEEE conference on Computer Vision and Pattern Recognition*, 1998.

[27] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, No. 11, 1222–1239, Nov. 2001.

[28] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods", *IJCV*, vol. 61, No. 3, 211–231, 2005.

[29] Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen, "Unstructured lumigraph rendering." In the proceedings of the *Proc. of SIGGRAPH 2001*, 425–432, 2001.

[30] Michael C. Burl, Markus Weber, and Pietro Perona, "A probabilistic approach to object recognition using local photometry and global geometry." In the proceedings of the *ECCV (2)*, 628–641, 1998.

[31] V. Cerny, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm", *Journal of Optimization Theory and Applications*, vol. 45(1), 41–51, 1985.

[32] T. Chan and J. Shen, "Non-texture inpaintings by curvature-driven diffusions", *J. Visual Comm. Image Rep.*, vol. 12(4), 436–449, 2001.

[33] S. Grace Chang, Bin Yu, and Martin Vetterli, "Spatially adaptive wavelet thresholding with context modeling for image denoising." *IEEE Transactions on Image Processing*, vol. 9, No. 9, 1522–1531, 2000.

[34] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "Approximation algorithms for the metric labeling problem via a new linear programming formulation", In the proceedings of the 12^{th} *Annual ACM-SIAM Symposium on Discrete Algorithms*, 109–118, 2001.

[35] Paul B. Chou and Christopher M. Brown, "The theory and practice of bayesian image labeling", *Int. J. Comput. Vision*, vol. 4, No. 3, 185–210, 1990.

[36] Antonio Criminisi, P. Pérez, and K. Toyama, "Object removal by exemplar-based inpainting." In the proceedings of the *CVPR*, 2003.

[37] Paul E. Debevec, C. J. Taylor, and Jitendra Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach", In the proceedings of the *SIGGRAPH 96*, 11–20, 1996.

[38] Paul E. Debevec, Yizhou Yu, and George Borshukov, "Efficient view-dependent image-based rendering with projective texture-mapping." In the proceedings of the *Rendering Techniques Eurographics Workshop*, 105–116, 1998.

[39] Gianfranco Doretto and Stefano Soatto, "Editable dynamic textures." In the proceedings of the *CVPR (2)*, 137–142, 2003.

[40] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun, "Fragment-based image completion", In the proceedings of the *SIGGRAPH*, 2003.

[41] Jaynes E., "On the rationale of maximum-entropy nethods." In the proceedings of the *Proceedings of the IEEE*, vol. 70, 626–633, 1982.

[42] Alexei A. Efros and William T. Freeman, "Image quilting for texture synthesis and transfer", In the proceedings of the *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, 341–346, 2001.

[43] Alexei A. Efros and Thomas K. Leung, "Texture synthesis by non-parametric sampling." In the proceedings of the *ICCV*, 1999.

[44] O. Faugeras, Q. T. Luong, and T. Papadopoulo, *The Geometry of Multiple Images : The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*, MIT Press, 2001.

[45] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Efficient belief propagation for early vision." In the proceedings of the *CVPR*, 2004.

[46] Pedro F. Felzenszwalb and Daniel P. Huttenlocher, "Pictorial structures for object recognition." *International Journal of Computer Vision*, vol. 61, No. 1, 55–79, 2005.

[47] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning", In the proceedings of the *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 264–271, june 2003.

[48] Mário A. T. Figueiredo, "Bayesian image segmentation using wavelet-based priors." In the proceedings of the *CVPR (1)*, 437–443, 2005.

[49] Andrew W. Fitzgibbon, "Stochastic rigidity: Image registration for nowhere-static scenes", In the proceedings of the *ICCV*, 662–669, 2001.

[50] S. Fleishman, B. Chen, A. Kaufman, and D. Cohen-Or, "Navigating through sparse views", In the proceedings of the *VRST99*, pp. 82-87, 1999.

[51] Daniel Freedman and Matthew W. Turek, "Illumination-invariant tracking via graph cuts." In the proceedings of the *CVPR (2)*, 10–17, 2005.

[52] Daniel Freedman and Tao Zhang, "Interactive graph cut based segmentation with shape priors." In the proceedings of the *CVPR (1)*, 755–762, 2005.

[53] William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael, "Learning low-level vision", *International Journal of Computer Vision*, vol. 40, No. 1, October 2000.

[54] S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images", *PAMI*, vol. 6, No. 6, 721–741, nov 1984.

[55] A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, 1985.

[56] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[57] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen, "The lumigraph", In the proceedings of the *SIGGRAPH 96*, 43–54, 1996.

[58] G.R. Grimmett, "A theorem about random fields", *Bulletin of the London Mathematical Society*, vol. 5, 81–84, 1973.

[59] A. Gupta and E. Tardos, "Constant factor approximation algorithms for a class of classification problems", In the proceedings of the *Proceedings of the 32^{nd} Annual ACM Symposium on the theory of Computing*, 652–658, 2000.

[60] R.I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge, 2000.

[61] David J. Heeger and James R. Bergen, "Pyramid-based texture analysis/synthesis", In the proceedings of the *SIGGRAPH*, 229–238, 1995.

[62] F. Heitz and P. Bouthemy, "Multimodal estimation of discontinuous optical flow using markov random fields", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, No. 12, 1217–1232, 1993.

[63] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin, "Image analogies", In the proceedings of the *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, 327–340, 2001.

[64] John H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[65] Berthold K. P. Horn and Brian G. Schunck, "Determining optical flow." *Artif. Intell.*, vol. 17, No. 1-3, 185–203, 1981.

[66] R.A. Hummel and S.W Zucker, "On the foundations of relaxation labeling process", *PAMI*, vol. 5, 267–286, 1983.

[67] Michal Irani, "Multi-frame optical flow estimation using subspace constraints." In the proceedings of the *ICCV*, 626–633, 1999.

[68] Hiroshi Ishikawa, "Exact optimization for markov random fields with convex priors." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, No. 10, 1333–1336, 2003.

[69] H. Ishikawa and D. Geiger, "Segmentation by grouping junctions", In the proceedings of the *IEEE conference on Computer Vision and Pattern Recognition*, 1998.

[70] Jiaya Jia and Chi Keung Tang, "Image repairing: Robust image synthesis by adaptive nd tensor voting." In the proceedings of the *CVPR*, 2003.

[71] Yedidia J.S., Freeman W.T., and Weiss Y, "Understanding belief propagation and its generalizations", In the proceedings of the *International Joint Conference on Artificial Intelligence*, Distinguished Lecture track, 2001.

[72] Steven L. Kilthau, Mark S. Drew, and Torsten Möller, "Full search content independent block matching based on the fast fourier transform." In the proceedings of the *ICIP (1)*, 669–672, 2002.

[73] Junhwan Kim, Vladimir Kolmogorov, and Ramin Zabih, "Visual correspondence using energy minimization and mutual information", In the proceedings of the *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, IEEE Computer Society, Washington, DC, USA, p. 1033, 2003.

[74] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science, Number 4598, 13 May 1983*, vol. 220, 4598, 671–680, 1983.

[75] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relaionships: metric labeling and markov random fields", *Journal of the ACM*, vol. 49, 616–630, 2002.

[76] R. Koch, "3d surface reconstruction from stereoscopic image sequences", In the proceedings of the *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV 1995)*, 109–114, 1995.

[77] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization", Technical Report MSR-TR-2005-38, Microsoft, 2005.

[78] Vladimir Kolmogorov and Yuri Boykov, "What metrics can be approximated by geo-cuts, or global optimization of length/area and flux." In the proceedings of the *ICCV*, 564–571, 2005.

[79] Vladimir Kolmogorov and Martin Wainwright, "On the optimality of tree-reweighted max-product message passing", In the proceedings of the 21^{st} *Conference on Uncertainty in Artificial Intelligence*, 2005.

[80] Vladimir Kolmogorov and Ramin Zabih, "Computing visual correspondence with occlusions via graph cuts." In the proceedings of the *ICCV*, 508–515, 2001.

[81] Vladimir Kolmogorov and Ramin Zabih, "Multi-camera scene reconstruction via graph cuts." In the proceedings of the *ECCV*, 82–96, 2002.

[82] Vladimir Kolmogorov and Ramin Zabih, "What energy functions can be minimized via graph cuts?" In the proceedings of the *ECCV*, 65–81, 2002.

[83] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra, "Texture optimization for example-based synthesis", *ACM Transactions on Graphics, SIGGRAPH 2005*, August 2005.

[84] Vivek Kwatra and *et al*, "Graphcut textures: Image and video synthesis using graph cuts", In the proceedings of the *SIGGRAPH*, 2003.

[85] David Lee and Theo Pavlidis, "One-dimensional regularization with discontinuities", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 10, No. 6, 822–829, 1988.

[86] M. Levoy and P. Hanrahan, "Light field rendering", In the proceedings of the *SIGGRAPH 96*, 31–42, 1996.

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

[88] X. Li and M. T. Orchard, "Spatially adaptive image denoising under overcomplete expansion", In the proceedings of the *Proc. IEEE ICIP*, 2000.

[89] Lin Liang, Ce Liu, Ying Qing Xu, Baining Guo, and Heung Yeung Shum, "Real-time texture synthesis by patch-based sampling." *ACM Trans. Graph.*, vol. 20, No. 3, 127–150, 2001.

[90] Michael H. Lin and Carlo Tomasi, "Surfaces with occlusions from layered stereo." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, No. 8, 1073–1078, 2004.

[91] Jun S. Liu, *Monte Carlo Strategies in Scientific Computing*, Springer Series in Statistics, 2001.

[92] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3D Vision*, Springer, 2005.

[93] Maurits Malfait and Dirk Roose, "Wavelet-based image denoising using a markov random field a priori model." *IEEE Transactions on Image Processing*, vol. 6, No. 4, 549–565, 1997.

[94] David R. Martin, Charless Fowlkes, and Jitendra Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, No. 5, 530–549, 2004.

[95] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan, "Image-based visual hulls", In the proceedings of the *Proc. of SIGGRAPH 2000*, 369–374, 2000.

[96] Leonard McMillan, *An Image-Based Approach to Three-Dimensional Computer Graphics*, Ph.D. thesis, University of North Carolina, Apr 1997.

[97] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering approach", In the proceedings of the *SIGGRAPH95*, pp. 39-46, 1995.

[98] N. Metropolis, A.W. Rceenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equation of state calculations by fast computing machines", *Journal of Chemical Physics*, vol. 21, 1087–1092, 1953.

[99] David Nistér, "Automatic passive recovery of 3d from images and video." In the proceedings of the *3DPVT*, 438–445, 2004.

[100] Faugeras O. and Berthod M., "Improving consistency and reducing ambiguity in stochastic labelling: An optimization approach", *PAMI*, vol. 3, 412–423, 1983.

[101] Sylvain Paris, François Sillion, and Long Quan, "A surface reconstruction method using global graph cut optimization", *International Journal of Computer Vision*, vol. 66, No. 2, 141–161, February 2006.

[102] K.A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting of occluding and occluded objects", In the proceedings of the *IEEE International Conference on Image Processing*, 2005.

[103] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[104] Aleksandra Pizurica, Wilfried Philips, Ignace Lemahieu, and Marc Acheroy, "A joint inter- and intrascale statistical model for bayesian wavelet based image denoising." *IEEE Transactions on Image Processing*, vol. 11, No. 5, 545–557, 2002.

[105] Tomaso Poggio, Vincent Torre, and Christof Koch, "Computational vision and regularization theory", *Nature*, vol. 317, No. 26, 314–319, 1985.

[106] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool, "Hand-held acquisition of 3d models with a video camera", In the proceedings of the *Proc. 3DIM'99*, 14–23, 1999.

[107] Javier Portilla and Eero P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients." *IJCV*, vol. 40, No. 1, 49–70, 2000.

[108] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli, "Image denoising using scale mixtures of gaussians in the wavelet domain." *IEEE Transactions on Image Processing*, vol. 12, No. 11, 1338–1351, 2003.

[109] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, 2nd ed., 1992.

[110] Christian P. Robert and George Casella, *Monte Carlo Statistical Methods (second edition)*, Springer Texts in Statistics, 2004.

[111] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations", *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, 420–433, 1976.

[112] Carsten Rother, Sanjiv Kumar, Vladimir Kolmogorov, and Andrew Blake, "Digital tapestry", In the proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[113] S. Roy and I. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem", In the proceedings of the *Proceedings of the International Conference on Computer Vision*, 492–499, 1998.

[114] Y. Wu S. Soatto, G. Doretto, "Dynamic textures", In the proceedings of the *Intl. Conf. on Computer Vision*, "439–446".

[115] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms", *International Journal of Computer Vision*, vol. 47, No. 1/2/3, 7–42, April-June 2002.

[116] Hartmut Schirmacher, Ming Li, and Hans Peter Seidel, "On-the-fly processing of generalized Lumigraphs", In the proceedings of the *Proc. of Eurographics 2001*, vol. 20, C165–C173;C543, 2001.

[117] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa, "Video textures", In the proceedings of the *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 489–498, 2000.

[118] Mark Segal and Kurt Akeley, "The OpenGL Graphics System: A Specification (Version 1.5)", http://www.opengl.org.

[119] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[120] Heung Yeung Shum, Richard Szeliski, Simon Baker, Mei Han, and P. Anandan, "Interactive 3d modeling from multiple images using scene regularities." In the proceedings of the *SMILE*, 236–252, 1998.

[121] Cyril Soler, Marie Paule Cani, and Alexis Angelidis, "Hierarchical pattern mapping." In the proceedings of the *SIGGRAPH*, 673–680, 2002.

[122] J. L. Starck, D.L. Donoho, and E. Candès, "Very high quality image restoration", In the proceedings of the *SPIE conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing IX, San Diego, 1-4 August*, edited by A. Laine, M. Unser, and A. Aldroubi, SPIE, 2001.

[123] Christoph Strecha, Rik Fransens, and Luc J. Van Gool, "Wide-baseline stereo from multiple views: A probabilistic account." In the proceedings of the *CVPR (1)*, 552–559, 2004.

[124] Christoph Strecha and Luc J. Van Gool, "Pde-based multi-view depth estimation." In the proceedings of the *3DPVT*, 416–427, 2002.

[125] Christoph Strecha, Tinne Tuytelaars, and Luc J. Van Gool, "Dense matching of multiple wide-baseline views." In the proceedings of the *ICCV*, 1194–1201, 2003.

[126] Jian Sun, Heung Yeung Shum, and Nan Ning Zheng, "Stereo matching using belief propagation." In the proceedings of the *ECCV (2)*, 510–524, 2002.

[127] Jian Sun, Lu Yuan, Jiaya Jia, and Heung Yeung Shum, "Image completion with structure propagation", In the proceedings of the *SIGGRAPH*, 2005.

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

[129] R. Szeliski, "Video mosaics for virtual environments", *IEEE CGA*, vol. 16, No. 2, 22–30, March 1996.

[130] Martin Szummer and Rosalind W. Picard, "temporal texture modeling", In the proceedings of the *Proc. of Int. Conference on Image Processing*, vol. 3, 823–826, 1996.

[131] S. Teller, "Automated urban model acquisition: Project rationale and status", 1998.

[132] Demetri Terzopoulos, "Regularization of inverse visual problems involving discontinuities", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, No. 4, 413–242, 1986.

[133] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song Chun Zhu, "Image parsing: Unifying segmentation, detection, and recognition", In the proceedings of the *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, IEEE Computer Society, Washington, DC, USA, p. 18, 2003.

[134] Zhuowen Tu and Song Chun Zhu, "Image segmentation by data-driven markov chain monte carlo", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, No. 5, 657–673, 2002.

[135] Matthew Uyttendaele, Antonio Criminisi, Sing Binb Kang, Simon Winder, Richard Hartley, and Richard Szeliski, "High-quality image-based interactive exploration of real-world environments", *IEEE Computer Graphics & Applications*, 2004.

[136] V. Vazirani, *Approximation Algorithms*, Springer, 2001.

[137] Sundar Vedula, Simon Baker, and Takeo Kanade, "Spatio-temporal view interpolation", In the proceedings of the *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002.

[138] O. Veksler, *Efficient graph-based energy minimization methods in computer vision*, Ph.D. thesis, Department of Computer Science, Cornell University, 1999.

[139] Olga Veksler, "Stereo correspondence by dynamic programming on a tree." In the proceedings of the *CVPR (2)*, 384–390, 2005.

[140] M. Wainwright, T. Jaakkola, and A. Willsky, "Map estimation via agreement on (hyper)trees: messagepassing and linear programming approaches", In the proceedings of the *Allerton Conference on Communication, Control and Computing*, 2002.

[141] J.Y.A. Wang and E.H. Adelson, "Layered representation for motion analysis", In the proceedings of the *CVPR93*, 361–366, 1993.

[142] Markus Weber, Max Welling, and Pietro Perona, "Unsupervised learning of models for recognition", In the proceedings of the *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, Springer-Verlag, London, UK, 18–32, 2000.

[143] Li Yi Wei and Marc Levoy, "Fast texture synthesis using tree-structured vector quantization", In the proceedings of the *Siggraph 2000, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 479–488, 2000.

[144] Yair Weiss, "Segmentation using eigenvectors: A unifying view." In the proceedings of the *ICCV*, 975–982, 1999.

[145] Yair Weiss and William T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs." *IEEE Transactions on Information Theory*, vol. 47, No. 2, 736–744, 2001.

[146] Yonatan Wexler, Eli Shechtman, and Michal Irani, "Space-time video completion." In the proceedings of the *CVPR (1)*, 120–127, 2004.

[147] Gerhard Winkler, *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*, Springer Verlag, 2003.

[148] Zhenyu Wu and Richard M. Leahy, "An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, No. 11, 1101–1113, 1993.

[149] J. Yedidia, W. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms", Technical Report TR2004-040, MERL, 2004.

[150] Ramin Zabih and Vladimir Kolmogorov, "Spatially coherent clustering using graph cuts." In the proceedings of the *CVPR*, 437–444, 2004.

[151] C. Lawrence Zitnick and Takeo Kanade, "A cooperative algorithm for stereo matching and occlusion detection." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, No. 7, 675–684, 2000.

[152] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski, "High-quality video view interpolation using a layered representation", *ACM Trans. Graph.*, vol. 23, No. 3, 600–608, 2004.

[153] D Zwilliger, *Handbook of Differential Equations*, Boston, MA: Academic Press, 1997.

Author's publication list

- N. Komodakis and G. Tziritas, "Image Completion Using Global Optimization", to appear in Proceedings of *CVPR 2006 (IEEE International Conference on Computer Vision and Pattern Recognition)*

- N. Komodakis and G. Tziritas, "Approximate Labeling via Graph-Cuts Based on Linear Programming", Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (under minor revision)

- N. Komodakis and G. Tziritas, "Morphable 3D-Mosaics: a Framework for the Visual Reconstruction of Large Natural Scenes", in video proceedings of IEEE Conference on Computer Vision and Pattern Recognition (VPCVPR), 2006.

- N. Komodakis and G. Tziritas, "A New Framework for Approximate Labeling via Graph Cuts", In Proceedings of *ICCV 2005 (IEEE International Conference on Computer Vision)*

- N. Komodakis and G. Tziritas, "Morphable 3D-Mosaics: a Hybrid Framework for the Visual Reconstruction of Large-Scale Outdoor Environments", (submitted)

- N. Komodakis, C. Panagiotakis and G. Tziritas, "3D Visual Reconstruction of Large Scale Natural Sites and Their Fauna", *Signal Processing: Image Communication*, Vol. 20, No. 9-10, pp. 869-890, Oct. 2005.

- N. Komodakis, G. Pagonis and G. Tziritas, "Interactive walkthroughs using morphable 3D-mosaics", *In Proceedings of 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, 2004.

- N. Komodakis and G. Tziritas, "Approximate Labeling via the Primal-Dual

Schema", *University of Crete (Computer Science Department)*, Technical Report CSD-TR-05-01, February 2005