SUCCESSIVE CONVEXIFICATION FOR CONSISTENT LABELING

by

Hao Jiang

M.S., B.S., Harbin Engineering University, 1995 and 1993

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2006

APPROVAL

Name: Hao Jiang

Degree: Doctor of Philosophy

Title of thesis: Successive Convexification for Consistent Labeling

Examining Committee: Dr. Ted Kirkpatrick
Chair

Dr. Ze-Nian Li, Senior Supervisor
Professor, Computing Science
Simon Fraser University

Dr. Mark S. Drew, Senior Supervisor
Associate Professor, Computing Science
Simon Fraser University

Dr. Greg Mori, SFU Examiner
Assistant Professor, Computing Science
Simon Fraser University

Dr. Ramin Zabih, External Examiner
Associate Professor, Computer Science
Cornell University

Date Approved: _____

Abstract

In this thesis, a novel successive convexification scheme is proposed for solving consistent labeling problems with convex regularization terms, i.e., convex metric labeling. Many computer vision problems can be modeled as such consistent labeling problems. The main optimization term, the labeling cost, however, is typically non-convex, which makes the problem difficult. As well, the large search space, i.e., formally the large label set, makes such applications thorny and inefficient to solve using traditional schemes. The proposed scheme successively convexifies the labeling cost surfaces by replacing them with their lower convex hulls, each time starting from the original cost surfaces but within shrinking trust regions, with a careful scheme for choosing new search regions. This enables the new scheme to solve a sequence of much easier convex programming problems and almost always find the correct labeling. The proposed scheme can be applied to labeling problems with any convex regularization terms. In particular, problems with L_1-norm regularization terms can be solved with sequential linear programming; and problems with L_2-norm regularization terms with sequential quadratic programming. To zero in on the targets in the search space, the method uses a set of basis labels to approximate the cost surface for each site, and this essentially decouples the size of the relaxed convex problem from the number of labels. The proposed scheme also has other useful properties making it well-suited to very large label-set problems, e.g. searching within an entire image. The proposed successive convexification scheme has been applied to many challenging computer vision problems: the task of robustly locating objects in cluttered environments, dense motion estimation with occlusion inference, appearance-adaptive object tracking with boundary refinement, and finally the challenging problem of human posture and action detection both in still images and in video. Compared with traditional methods, the proposed scheme is shown to have a clear advantage in these applications.

To my Mom and Dad.

Acknowledgments

I would like to thank my supervisors Dr. Ze-Nian Li and Dr. Mark S. Drew. Without their advising, support and encouragement, the thesis work cannot be done. I also would like to thank Dr. Ramin Zabih and Dr. Greg Mori for valuable comments to improve the thesis. I want to thank my family and friends for their persistent encouragement and help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Regularized energy minimization and consistent labeling can be used to model a variety of computer vision problems. For instance, motion estimation can be modeled as a consistent labeling problem in which we assign labels (the motion vectors) to sites (pixels) in an image so that a properly designed energy function is minimized. The consistency of label assignment for nearby sites must be enforced, and therefore smoothing or regularization terms must be included in the energy function. Many other vision problems such as object detection, image reconstruction, image denoising, tracking and action recognition can also be formulated as consistent labeling problems. Unfortunately, consistent labeling in general is an NP-hard problem. It is a big challenge to solve these optimal labeling problems in computer vision because they usually have very large label set. Even though different methods have been studied, problems with very large label set are still largely unsolved. The question is "Is there any systematic method to simplify the searching space to make the searching largely decoupled from the size of the label set?" In this thesis, we show that such schemes can be constructed for consistent labeling problems with convex regularization terms. The idea of simplifying the target label space enables us to develop a novel consistent labeling scheme — *successive convexification*. We explore this scheme in detail and devise new methods based on successive convexification to solve hard computer vision problems. The proposed methods yield better results and are more efficient than previous schemes. Using the successive convexification scheme, we consider applications including object detection, large scale motion estimation, appearance adaptive object tracking with boundary refinement, and human activity recognition.

1.1 Consistent Labeling and Regularized Optimization

Assuming that S and L are the site set and label set respectively, in consistent labeling we would like to optimize label assignment f_s for each site s so that an energy function is minimized:

$$\min\{\sum_{s\in S} c(s, f_s) + \sum_{\{p,q\}\in neighbors} d_1(f_p, f_q) + \sum_{\{x,y,z\}\in neighbors} d_2(f_x, f_y, f_z) + ...\} \qquad (1.1)$$

where $c(s, f_s)$ is the cost of assigning label f_s to site s; the other terms are regularization terms which penalize large discrepancies of labeling for nearby sites. Consistent labeling is thus also a regularized optimization problem. For computer vision applications, consistent labeling and regularized optimization are often equivalent. Here, we focus on the consistent labeling problems with convex regularization terms, for which $d_i(.)$ are convex functions. In its simplest form, a consistent labeling problem only has labeling cost terms and first order regularization terms. Higher order terms could appear in image reconstruction and contour tracking applications. The energy function of a consistent labeling problem is usually highly non-convex because of the non-convexity of the labeling cost function $c(.)$, which makes it very hard to solve using traditional methods, especially for large label set problems. We devise new methods to solve such problems more robustly and efficiently.

Optimizing consistent labeling also has a statistical explanation. From the statistics perspective, optimal labeling corresponds to an inference problem of finding a maximum posterior labeling configuration given the labeling cost and site topology. A labeling problem can be modeled as an inference problem on a Markov Random Field (MRF). Using graphical representation, in MRF, each node represents a random variable corresponding to a site on the template; the links in the graph show the conditionally dependent relations between the random variables — a random variable in a Markov Random Field is only conditionally dependent on its direct neighbors. It has been proved that for MRF, the joint distribution can be factorized into a series of productions involving random variables in the cliques of the MRF graph, i.e.,

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c\in \text{All Cliques}} \psi_c(\mathbf{x}_c)$$

where $\mathbf{x}$ is the collection of all the random variables and $\mathbf{x}_c$ is collection of random variables in a clique c (a clique is a fully connected sub-graph); $\psi_c(\mathbf{x}_c)$ is a potential function and Z

is a normalization factor. Usually $\psi_c(\mathbf{x}_c)$ can be defined based on another energy function

$$\psi_c(\mathbf{x}_c) = \exp\{-E(\mathbf{x}_c)\}.$$

The joint distribution can thus be written as

$$p(\mathbf{x}) = \frac{1}{Z}\exp\{-[\sum_{s\in S}\psi_{c_1}(x_s) + \sum_{\{p,q\}\in neighbors}\psi_{c_2}(x_p,x_q) + \sum_{\{i,j,k\}\in neighbors}\psi_{c_3}(x_i,x_j,x_k) + ...]\}.$$

The goal of consistent labeling is to find a configuration of $\mathbf{x}$ such that joint distribution $p(\mathbf{x})$ is maximized. Denoting $\psi_{c_1}(x_s)$ by $c(s,\ x_s)$ and $\psi_{c_2}(x_p,x_q)$ by $d(x_p,x_q)$, and so on, maximizing $p(\mathbf{x})$ is thus equivalent to minimizing the labeling energy in Eq. (1.1).

1.2 Successive Convexification for Labeling Problems

As discussed above, many important tasks in computer vision can be mathematically formulated as a consistent labeling problem, to assign labels to sites such that a predefined energy function is minimized. For consistent labeling problems, labels are usually defined in a metric space, yielding a label distance measure. Although simple in concept, consistent labeling is NP-hard [23]. It is so hard that in its general form, there is even no approximation schemes upper-bounded by a constant factor [69]. For some special cases, for instance, when sites have linear or tree order, dynamic programming [2] can be used to solve the labeling problem in polynomial time. Another special case is when labels for each site have linear order, and the metric defined in the label space is convex. In this case, polynomial-time max-flow schemes [4] [34] can be applied. Other searching schemes, e.g. branch and bound schemes [16], whose worst and average complexities are exponential, have also been applied to medium-size labeling problems. For general cases, approximation algorithms are preferred. Relaxation labeling (RL) [17] is one of the earliest methods for solving labeling problems, and has had a great deal of influence on later schemes. RL uses local search, and therefore relies on a good initialization. Iterative Conditional Modes (ICM) [3] — another widely applied method for solving labeling problems — is greedy, and has been found to be easily trapped in a local minimum. In recent years, Graph Cut (GC) [7] and Belief Propagation (BP) [20] [5] [6] have become popular methods for solving consistent labeling problems in vision. Graph Cut has been successfully applied in stereo [33], motion estimation [36], and segmentation [26]. Loopy Belief Propagation has also been widely applied in problems such as stereo [15] and object matching [9]. GC and BP are more robust than traditional

labeling schemes and are also found to be faster than methods based on stochastic annealing [27]. Besides GC and BP, many other schemes have been presented, such as spectrum graph theory based methods [18]. One of the serious problems for traditional labeling schemes is that their complexity increases rapidly with respect to the number of labels, e.g., BP has square complexity with respect to the number of labels. Robust methods such as GC and BP become slow for solving large scale problems that involve a large number of labels. To solve this problem, we need to carefully utilize the specific structure of consistent labeling problems.

Although intensively studied, large scale labeling problem is still unsolved. The difficulty is essentially due to the non-convexity of the labeling cost function and very large set of labels. In this thesis, we study an efficient scheme for solving consistent labeling problems with convex regularization terms. Although no proof is found about the NP-hardness of such labeling problems, they are very likely to be so. A *successive convexification* (SC) scheme is proposed to solve these labeling problems Different from other methods in which all the candidate labels are involved in the solution process, the proposed scheme uses a much smaller number of basis labels to represent the label space and also progressively approaches the optimum. These components of the method greatly speed up the algorithm. In our scheme, *basis labels* correspond to the vertex coordinates of the $(n+1)$-D lower convex hull of the cost surface associated with each site. (E.g., for matching x, y points, $n = 2$.) Since the basis labels are determined by the label cost surface shapes and only weakly related to the density of sampling for discrete labels, the size of the convex program is mostly decoupled with the number of labels. We propose the successive convexification scheme to iteratively increase the accuracy of the approximation. During the iteration, we shrink the trust region for each site and locate a new trust region based on the solution of the previous relaxation, but re-convexify the *original* cost function in the refined search region. This process continues until the trust region becomes small. Since the convexification process eliminates many false local minima in the earlier stages of the solution process, the proposed scheme is able to find a good approximated solution quickly. Iteratively, the successive relaxation process refines the labeling result.

The work most related to the proposed scheme is the mathematical programming schemes, which have received much interest in formulating and solving labeling problems. The early RL schemes belong to this class. A major challenge in algorithm development is overcoming the problem of local minima in the optimization process, and different schemes have

been proposed. Deterministic annealing schemes [10] [19] have been successfully applied to matching point sets and graphs. Quadratic programming [21] and most recently semidefinite programming [22] have also been proposed for matching, a standard consistent labeling problem. To date, these schemes have only been usefully applied to small scale problems. Linear Programming (LP) has also been applied in many vision problems, such as in estimating the motion of rigid scenes [24]. A linear programming formulation [23] has been presented for uniform labeling problems and for approximating general problems by tree metrics. Another general LP scheme, studied in [13], is quite similar to the linear relaxation labeling formulation [17]. This LP formulation is found to be only applicable to small problems because of the large number of constraints and variables involved. Another major problem for traditional convex programming relaxation schemes is that they try to solve the labeling problem by a single relaxation, which is usually followed by a rounding process. For complex vision problems, a single convex relaxation is usually not sufficient to capture the highly nonlinear nature of the problem.

The proposed successive convexification scheme is different from the well-known GNC (graduated non-convexity) scheme. GNC is a special case of the continuation method [90]. GNC has been used to convexify objective functions [11] where non-convexity is caused by explicit non-convex functions, e.g. the truncated L_2 distance function. GNC replaces non-convex functions in the first iteration with approximated convex functions and then gradually restores these functions into their original shapes. GNC has also been used to convexify feasible regions for graph matching [12], where the objective functions are still non-convex. For labeling problems with non-parametric labeling cost terms, e.g. image matching, GNC has not been used for convexifying objective functions due to the difficulty in convexifying non-parametric labeling cost surfaces. We develop our successive convexification scheme to convexify the non-parametric cost terms, making it amenable to solution by robust convex programming routines. In successive convexification, we shrink the trust region for each site and reconvexify the surfaces in these smaller trust regions.

The proposed convexification and trust region shrinking scheme is in fact quite general, and can be used to improve the results of many mathematical programming based schemes. For example, a consistent labeling problem with L_2 regularization terms can be solved using successive convexification and convex quadratic programming. And, linear programming relaxation can be used to solve problems involving L_1 regularization terms. The proposed successive convexification scheme has been applied as a core technique to different computer

vision applications.

In the following, we analyze current methods for consistent labeling in computer vision with more details.

1.2.1 Exact Schemes for Consistent Labeling

In its general formulation, consistent labeling problem is *NP*-hard. For some special cases, efficient algorithms do exist. For instance, when an ordering of the sites is available, a dynamic programming (DP) method can be used to solve the optimal labeling problem. In the simplest case, sites have a linear order and the standard Trellis method can be applied. Dynamic programming is an efficient way for exhaustive search. It has high temporal and spatial complexity and therefore is usually used in relatively small scale problems. Dynamic programming can also be applied to problems in which site topologies are trees. For such problems, each subtree can be simplified into a single node recursively. Simple branch (a linear graph) can be directly optimized by a standard Trellis algorithm. We repeat the process from the bottom of the tree until it goes to the root node. Reverse path following is then carried out to label each node in the tree. For cases when there is no explicit linear or tree structure for sites, some constructive procedure can be applied so as to generate structures suited for dynamic programming [2]. Dynamic programming cannot deal with cases when there are cycles in site topology. But we can use dynamic programming in the case of a single cycle. By splitting the circle, forming a line structure and at the same time constraining the label assignment be the same for the head and the tail, we then can apply standard dynamic programming. DP has been used in stereo [61] and object recognition [74]. Its inability to deal with general graphs such as grids causes artifacts such as line-by-line inconsistency in stereo.

Another special case for which an exact solution exists is that if the labels for each site have a linear order, problems with convex regularization terms can be solved by maximum flow schemes [4]. The maximum flow method converts a consistent labeling problem into finding a minimum cut in a properly designed graph. If capacities of edges are properly set, the minimum cut also minimizes the labeling energy function. Maximum flow schemes have been applied to stereo [4], object segmentation [62] and tracking [63]. A compound graph structure [64] is also presented to solve multi-layer contour matching problems. The condition that the labels must have a linear structure restricts the applications of max-flow method in general consistent labeling problems.

The above exact labeling methods have polynomial complexity. For medium scale problems, exact exponential complexity algorithm can also be used. A concave programming method [66] is proposed to find a permutation matrix P to minimize the matching cost $J(P, C) = trace(PC)$. The constraint for P is that the summation of its rows and columns are both not greater than 1 and the elements of P are either 0 or 1. To prevent the trivial solution $P = 0$, another constraint $\sum p_{ij} = M$ is set. C is a cost matrix, with element C_{ij} the cost of matching site i with target j. This integer programming can be converted to an equivalent concave programming problem, which gives exact solution. Branch and bound for solving integer programming problems has also been applied to feature point matching problems [16]. Branch and bound method solves a relaxed problem in which the variables are turned from integers to floating point numbers. Each non-integer variable generates two more constraints and new relaxed mathematical programming problems in each branch are solved; the result is then compared with an upper bound of the problem: if the objective function is greater than the upper bound, the branch is cut. Branch and bound is an efficient way to do exhaustive search and therefore gives exact solutions. The complexity of the algorithm is high and not suited for large scale problems.

1.2.2 Approximation Methods

For most vision problems, however, these conditions about the site topology and target label structure do not hold. Cycle free topology is also found not to be sufficient for many vision problems [77], even though they can be globally optimized using dynamic programming. In general situations, the optimal labeling problem is a hard problem (or at least it is unknown whether there are efficient algorithms). Therefore, we have to resort to other searching schemes and usually we can only obtain an approximation solution. These searching methods can be classified into two categories: greedy methods and global searching methods.

Greedy Methods Greedy schemes, also known as local optimization methods, have been intensively studied in different computer vision problems. The basic idea of local optimization is that starting from an initial position, the solver tries to locate a better solution nearby. Such a process iterates until the solution cannot be further improved. Local optimization does not intend to find a globally optimal solution and usually works well only when the initial value is near the true optimum. In computer vision research, a large class

of methods based on variational calculus and Partial Differential Equations (PDE) are local optimization methods. Variational and PDE methods are very rich in their ability to formulate and analyze complex problems. Usually, the gradient descent, a standard local optimization method, is used in the solution procedure. PDE methods have been widely applied to image denoising [70] , image restoration [71], active contours [44], optical flow [29], and 3D surface reconstruction [65]. The drawback of PDE schemes is their local optimization property which makes the quality of results greatly rely on initial values. For discrete labeling problems, one of the most commonly used greedy schemes is ICM [3]. ICM does exhaustive search for each site while keeping the labels of other sites unchanged. For each site, the label achieving the lowest energy is chosen. Such a process iterates. ICM usually converges very quickly. Typical iteration steps are 20–50. But ICM is well known to be sensitive to the initial values and easily trapped in a local minimum.

Globally Searching Methods To overcome the problems of local optimum, researchers have studied different methods trying to find a global optimum solution. These schemes intend to obtain globally optimal solutions without relying on the initial values setting.

Simulated annealing, an early global optimization method, accepts a labeling configuration that increases energy with some probability and therefore enables the algorithm to overcome local minimums. This probability is characterized by an artificial temperature T that is high at the early stages and diminishes as the iteration goes on. This process is analogous to an annealing process in which a crystal forms. Theoretically, simulated annealing is always able to globally optimize a labeling objective function. But the process may take infinite length of time. Simulated annealing is found to be slow in most real applications. In recent years, several elegant new methods have been proposed.

Boykov and Zabih [7] develop the Graph Cut scheme that can be used in general labeling problems by applying a binary graph cut method iteratively. Label changing schemes – $\alpha\beta$ swap and α expansion – can be used in the iteration. We review the α expansion scheme here since it is one of the most commonly used methods in computer vision literature. In α expansion, all the sites are initialized with randomly selected labels. Then, the algorithm goes through an iteration procedure. At each iteration of α expansion, a random label is selected and denoted as α; and a binary decision has to be made for each site about whether the label for a site should be changed to α or kept unchanged. This binary decision can be globally optimized by finding the min-cut of an appropriately constructed graph. The

graph has to be reconstructed in each iteration of the α expansion. The iteration continues until there are no further changes of labeling.

The graph is constructed with the following scheme. First, nodes are generated for each site. One extra source node s and one extra sink node t are also constructed. For each neighboring pair of nodes p and q, a sub-structure as Fig 1.1 is constructed. The structure is dependent on the current status of the label assignment. If two neighboring sites p and q have the same label β and $\beta \neq \alpha$, the sub-graph is constructed as Fig 1.1 (a). The values beside each edge indicate the capacities of the corresponding edges. In this illustration, the cut has value $C(p, \alpha) + C(q, \beta) + d(\beta, \alpha)$, where $C(x, y)$ is the cost of labeling site x with label y and $d(\beta, \alpha)$ is the distance between labels α and β. Therefore, the cut in the example equals the energy of labeling p with α and q with β in the consistent labeling problem involving two sites with pairwise regularization terms. It is not hard to verify that each cut corresponds to a labeling configuration. The decision of changing a site's label to α is made if the site is not reachable from the source after the min-cut is found; otherwise the site keeps its previous label. In another case when p and q have different labels, the graph structure is shown as Fig 1.1 (b). To guarantee the equivalence of min-cut and minimum labeling assignment energy, we need to enforce that min-cut with two d edges should not exist. Such a condition can be guaranteed by constraining

$$d(x, y) + d(y, z) > d(x, z).$$

To construct the whole graph involving more than two sites, we just need to repeat the subgraph construct procedure and merge the subgraphs together by merging corresponding nodes and edges. In cases of α equaling some site's label, we can issue an infinite capacity edge from the source node to that site node.

It is not difficult to show that the Graph Cut indeed optimizes the binary decision. When multiple passes of the procedure apply, the Graph Cut becomes an efficient global searching scheme. Other energy formulations [8] can also be minimized by the graph cut framework. The Graph Cut has also been applied to many labeling problems in vision [7] [8].

The Graph Cut is a deterministic labeling algorithm. Another recent method, Belief Propagation, studies the labeling problems from the statistics point of view. Optimal labeling is obtained by propagating beliefs between nodes of graphical models. For linear or tree structure MRF, there is an efficient exact inference algorithm [20]. In our labeling

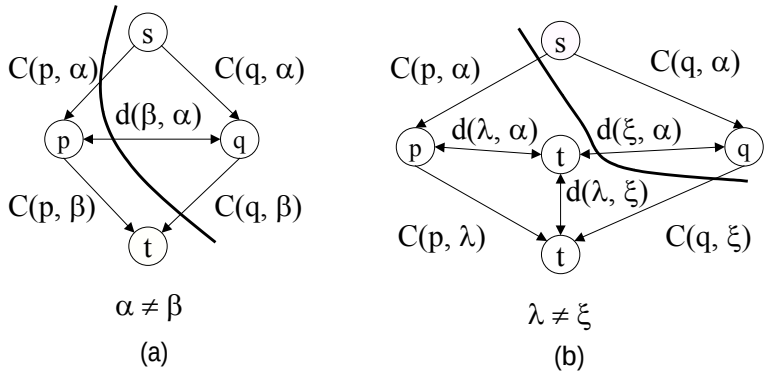


Figure 1.1: Graph structures in Graph Cut.

problem, the algorithm is called the max-product algorithm. For linear or tree topology, max-product algorithm is equivalent to dynamic programming. The max-product algorithm can be interpreted as propagating beliefs between nodes and therefore often denoted as belief propagation. The message passing framework can be generalized into arbitrary graphs and gives exact inference. This procedure is known as the junction tree algorithm [72]. For large scale problems, this algorithm generates many variables and becomes too slow to be applied in practical applications. Loopy belief propagation [5] is an approximation method which directly applies the message passing procedures in trees to general graphs with loops.

For loopy graphs with only pairwise relations, belief updating process is as follows. We assume that the labeling energy function is written as

$$U(x) = \sum_{s \in S} c(s, x_s) + \sum_{(p,q) \in \mathcal{N}} d(x_p, x_q).$$

Denoting m_{pq}^t to be the messages p sending to its neighbor q at iteration t, the belief updating process is as follows:

$$m_{pq}^t(x_q) = \min_{x_p} \{ d(x_p, x_q) + c(p, x_p) + \sum_{s \in \mathcal{N}(p) \setminus q} m_{sp}^{t-1}(x_p) \}$$

All the entries in m_{pq}^0 are initialized to be zero. After T iterations, a belief vector is calculated for each node:

$$b_q(x_q) = c(q, x_q) + \sum_{p \in \mathcal{N}(q)} m_{pq}^T(x_q)$$

The label x_q^* that minimizes $b_q(x_q)$ is selected as the label for site q. For general graphs with loops, the message passing in BP may go through a node more than once. And, there is no natural method to determine the termination point. Even though there is no guarantee that BP converges for general graph topology, it is found to work well in many real applications, e.g., stereo [15] and object matching [9] etc. An efficient BP scheme [6] is presented for problems in which sites are formed into grids.

Mathematical Programming Schemes Mathematical programming schemes are more general methods for solving consistent labeling problems. Modern optimization methods such as convex programming provide many powerful tools for solving complex optimization problems. Formulating labeling problems such that we can fully utilize the power of these methods is a challenge in mathematical programming based schemes. Here we illustrate some related work with the proposed framework.

Chui and Rangarange [10] studied a deterministic annealing method for point set matching problems. The energy function also contains two terms: one is determined by the Euclidean distance of the warped source points and the target points, and the other is a regularization term smoothing the second order derivative of the warping. The formulation can be reformulated using thin-plate-spline warping that combines an affine mapping term and an extra non-linear warping term. The difficulty of the problem lies in that there is no knowledge of the matching between the source and target point sets. Chui and Rangarange use an alternate method to estimate the matching and warping. The deterministic annealing procedure sets relatively small weight on the non-integral penalty term for the label assignment coefficients (they can be float numbers in 0-1) at the early iteration stages. The weight of the integral constraint increases and finally assignment coefficients become 0 or 1.

Linear programming formulation for consistent labeling problem has also been presented [13]. The basic idea is that we define a binary variable $x(p, i, q, j)$ to represent whether site p is assigned a label i and q is assigned a label j; the statement is true if $x(p, i, q, j)$ is 1 and otherwise 0. Another variable $y(s, i)$ is 1 if site s is assigned a label i and otherwise 0. In fact these two kinds of binary variables are correlated and $\sum_{j \in L_q} x(p, i, q, j) = y(p, i), \forall \{p, q\} \in \mathcal{N}$, where $\mathcal{N}$ is the neighboring site pair set, and L_q is the label set for site q. We can then use these binary variables to convert the original consistent labeling problem into an integer programming problem written as

$$\min \sum_{s \in S} \sum_{i \in L_s} c(s,i) \cdot y(s,i) + \sum_{\{p,q\} \in \mathcal{N}} \sum_{i \in L_p} \sum_{j \in L_q} d(p,i,q,j) \cdot x(p,i,q,j)$$

$$\begin{aligned} s.t. \qquad \sum_{i \in L_s} y(s,i) &= 1, \forall s \in S \\ \sum_{j \in L_q} x(p,i,q,j) &= y(p,i), \forall i \in L_p, \forall \{p,q\} \in \mathcal{N} \\ x(p,i,q,j) &= x(q,j,p,i) \\ y(s,i) \text{ are 0 or 1}, \qquad & x(p,i,q,j) \text{ are 0 or 1} \end{aligned}$$

Because of the hardness of solving integer programming problems, the binary constraint is discarded and the relaxation becomes linear programming with non-negative variables. This linear program is found to yield better bounds for optimization problems involving different norms. In real applications, it can only be used to solve small scale problems because of its very large number of constraints and variables. Berg et al. [73] present a LP relaxation scheme for consistent labeling problems formulated as integer quadratic programs. A linear program is formulated for lower-bounding the quadratic programming problem and a local search is used to refine the final solution. This method is applied to object class recognition problems.

Semidefinite programming has also been studied for labeling problems. Semidefinite programming is an extension of linear programming in which the variables form a matrix instead of a vector as in linear programming. Analogous to standard linear programming formulation in which the vector elements are greater than or equal to 0, the matrix in semidefinite programming is forced to be semidefinite. A standard method to formulate a semidefinite programming for labeling problems is by first formulating an integer quadratic programming problem.

$$\min\{c^T \mathbf{x} + \mathbf{x}^T D \mathbf{x}\}$$

$$s.t.\ A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{x} \text{ are 0s or 1s}$$

in which $\mathbf{x}$ are binary variables indicating assignment status of each label and site pair. We augment the vector of $\mathbf{x}$ into $\mathbf{y} = [\mathbf{x}^T, \mathbf{1}]^T$ and let $X = \mathbf{y}^T\mathbf{y}$. Now the integer quadratic programming can be rewritten using X:

$$\min C \cdot X$$

$$s.t.\ E_m \cdot X = d_m,\ X \geq 0,\ X \text{ is rank one}$$

where $\cdot$ is the dot product for matrices, defined as the summation of the product of corresponding elements of two matrices. We do not need to set binary constraints. Instead, we add constraints $x_i^2 - x_i = 0$ and X is a rank 1 semidefinite matrix. We discard the rank 1 constraint and the problem becomes a semidefinite programming problem. Semidefinite programming can be solved using prime-dual methods. Currently, semidefinite programming can only be used for small scale problems.

In summary, consistent labeling problems in computer vision with large label sets are still not solved. It is important to study these problems because many computer vision systems have to rely on robust labeling schemes to work well in real world situations. In this thesis, we propose new methods for robust and efficient labeling in convex metric spaces. The proposed successive labeling scheme searches the target point space differently from previous methods and is suited for very large label set problems. We further demonstrate the usefulness of the proposed scheme for different computer vision problems. We extend the labeling formulation to include outlier/occlusion inference in motion estimation and we show how to apply the proposed successive convexification scheme to this problem. Tracking can be treated as a sequential labeling problem in which we sequentially estimate target point assignment to template objects. We study robust successive convexification based method to solve this problem. We further apply the successive convexification scheme to posture and action detection in images and videos. Still posture recognition can be converted to a shape matching problem and the proposed successive convex relaxation scheme can be directly applied. For action detection, we extend the basic labeling problem to include inter-frame position constraint and we use successive convex relaxation method to solve this video sequence registration problem. The above applications involve large search spaces and are hard to solve with traditional schemes. The proposed successive convexification scheme can be used to greatly simplify the searching problem without sacrificing the qualities. The proposed schemes are faster and more robust for these difficult problems. In the following, we introduce these applications and related work.

1.3 Optimizing Motion Estimation

Motion estimation is a key technique for video processing and multimedia applications such as motion compensation for video compression, 3-D scene reconstruction and object

segmentation.

The motion of an object in space often leads to color displacement in images. By estimating the displacement of pixels in images, we can infer the movement of objects. In reality, it is usually not this simple. Movement of objects sometimes does not generate color changes at all, for example, because of occlusion; and conversely the color changes in images can be caused by illumination but not motion. To simplify the problem, constant brightness is usually assumed. If pixel displacements are small, optical flow based methods can be applied to motion estimation. In optical flow methods, color constant equation $I(x + dx, y + dy, t + dt) - I(x, y, t) = 0$ is approximated with $I_x dx + I_y dy + I_t dt = 0$ by keeping only the linear terms in Taylor expansion series. Letting u and v represent motion in the x and y directions respectively, we have the optical flow equation

$$I_x u + I_y v + I_t = 0$$

When estimating motions, simply using optical flow equation is not sufficient. After introducing a regularization term, Horn and Schunck [29] formulate optical flow as a variational problem and an iterative steepest descent scheme is used in optimization. Another well-known method to calculate optical flow is that of Lucas and Kanade [30], in which a weighted least square norm is used to fit the local first order optical flow constraint. There are also many other variants of optical flow methods based on the optical flow equation. These methods are carefully evaluated by Barron, et al [31]. Barron classified optical flow methods into four categories: differential method, correlation based method, phase based method and energy based method. A more recent optical flow method based on the structure tensor and a local parameter model is studied in [32]. Optical flow was originally designed for small scale motion estimation in which the low order (first or second order) approximation is adopted. Therefore, it cannot be directly used for estimating large motions. Multi-resolution schemes extend optical flow schemes such that they can be applied in larger scale motion estimation settings. The idea is to estimate motions in a coarse to fine fashion. Large motions become small motions in a coarse scale and therefore optical flow method is likely to succeed. Nevertheless, the motion searching range in optical flow methods is still very restricted. Usually the motion cannot be bigger than 1/4 of the size of the image.

To overcome the limitations of optical flow methods, we need to estimate motion by directly finding the pixel correspondence of successive video frames. In this approach, motion estimation corresponds to finding a mapping from a reference image to a target image, and

there is usually no low order approximation process involved in formulating the motion cost function. Thus this approach can inherently be applied to large scale motion. In some correspondence schemes, the mapping can be restricted to some specific parameter model, such as projective or affine motion estimation where the motion is modeled as a projective or affine transformation. For such problems, Hough transform or Geometric Hashing can be used to estimate the relatively small number of transformation parameters. Another kind of mapping is non-parametric, in which there is no explicit parametric motion constraint and can represent wider motion types. Non-parametric motion estimation has attracted a great deal of interest in recent years. Several matching schemes of horizontal motion estimation for rectified images in the stereo problem have been intensively studied [7, 33, 34, 4, 15]. Also, the Graph Cut [36] and a max-posterior probability based matching method [35] are presented for motion estimation for both horizontal and some 2-D motion problems.

In this thesis, we follow the image correspondence scheme and propose a convex programming based method [14] to estimate *large scale motion* for two images. There is no restriction of the topological structure of neighboring sites. Here, motion can have both x and y freedom in the two images. Our formulation is non-parametric and suitable for solving very large scale problems, involving several thousand sites, where the searching range for each site can be in a range as large as hundreds of pixels in both x and y directions. The proposed method can also estimate the occlusion and motion at the same time. The unique feature of the proposed scheme is that it successively convexifies the matching cost function in a sequence of focus regions. The successive convexification converts the original hard non-convex optimization problem into a sequence of convex problems which can be solved efficiently. After the convexification process, the motion vectors in the search region can be represented as the linear combination of a set of basis motion vectors. The proposed optimization scheme efficiently explores the search space by checking the triangles formed by these basis vectors. This scheme is much different from the searching procedure in standard schemes such as ICM, BP or GC and is more robust when the searching range becomes very large. Experiment shows that the proposed scheme has less matching errors than other methods such as ICM, BP and GC in matching ground truth testing data for large scale motion estimation. We further study a detail-preserving partial differential equation (PDE) method to estimate a refined dense motion map based on the initial matching from the successive convexification. Such a two-step scheme combines the strength of global searching and local detail searching and therefore is able to yield high quality motion estimation

efficiently.

1.4 Optimizing Object Tracking

We further studied robust object tracking schemes based on the proposed successive convexification method. Tracking is the process of repeatedly locating one or multiple objects in a sequence of video frames. Tracking objects in video has been an important task in computer vision systems. The ability to track objects is usually the first step in behavior recognition. Object tracking is deemed a hard problem, especially when the illuminance and object's appearance change dramatically.

Designing object tracking systems usually involves two problems. The first one is object representation and matching scheme. The second is dynamic model. Both problems have been intensively studied. In this thesis, we focus on the first problem in tracking. The simplest method of representing an object is using an image patch of the whole object [85]. Usually the template has a rectangle shape. If only translation is involved and the movement is small, we can use gradient descent method to track an object. Another method represents an object with "blobs" [86], which only represent the gross appearance of an object using Gaussian Mixture Model (GMM) in a joint space of color and position. Object tracking in videos becomes finding pixels that maximize the GMM. "Mean shift" [87] extends the "blobs" method and allows progressively updating an object's position without visiting all the pixels in each video frame. Instead of modeling foreground pixels, background models [88] have also been studied. Active contour or the snake [42] is another object representation in tracking. Optimizing active contour can be done based on partial differential equation schemes or dynamic programming.

In this thesis, we propose an appearance adaptive object tracking scheme [60] based on successive convexification. Objects are represented as meshes and we formulate mesh tracking as a sequential graph matching problem that can be solved robustly with the proposed successive convexification scheme. The proposed method is able to track objects with large rotation and scale changes. It also greatly reduces the drifting problem and thus works for very long video sequence. The proposed tracking method can also be used to track objects with drastic shape and appearance changes, due to severe viewpoint and aspect changes and object shape deformation.

Traditional appearance adaption methods can be classified into two classes. The first

class of methods use features resistant to object aspect and deformation, such as the color histogram. The mean-shift based method [76] belongs to this class. Methods relying on invariant features cannot handle large appearance changes and usually do not produce an accurate correspondence in tracking. The second class of methods are more correlated with the proposed method, which are appearance-based and require a set of key templates representing an object's appearance. Black and Jepson [67] propose a PCA based approach to select and represent templates. Researchers also study methods to learn the templates online [68].

We present an appearance adaptive object tracking scheme which does not rely on a complex training process. The system requires a very small number of exemplars. Graph templates are then generated and used in tracking based on the successive convexification scheme. The proposed scheme is able to track an object that changes appearance dramatically by selecting the best template in matching. To further increase efficiency, the templates are organized into a digraph, so that one template can only be replaced by its scaled or rotated version or its neighbors with the same scale and rotation settings. The tracking process is then equivalent to finding a node transition sequence in the given digraph. Experiments show very promising results for tracking objects in cluttered backgrounds. We also study a method to combine contour matching and mesh matching for boundary refinement. This problem can also be formulated as an energy minimization problem that can be solved with the proposed successive convex programming method.

1.5 Optimizing Posture and Action Detection

We finally study human posture and action detection based on the proposed successive convexification scheme. Recognizing human posture in images and videos is an important task in many multimedia applications, such as multimedia information retrieval, human computer interaction, and surveillance. Here, posture is defined as a snapshot of human body configurations. A sequence of postures can then be combined together to generate meaningful gestures. In many cases, a posture in one single image also conveys meaningful information. For example, it is possible for a human observer to disambiguate actions such as walking, running, standing, sitting, etc., from just a single image. In recent years, recognizing human body postures in general images or videos with a good deal of confounding background clutter has received much interest.

Recognizing human body configuration in controlled environments has been intensively studied in many experimental and commercial systems; to name a few: MIT Media Lab's `KIDSROOM` [46], `ALIVE` [47], Emering et al.'s gesture recognition system [48] and Vivid Group's gesture recognition system [49] aimed at HCI applications. These systems rely on segmentation of human objects from the background in a specific, restricted environment (the KIDSROOM, ALIVE, Vivid group's system etc.) or by position/velocity sensors attached to human subjects [48]. To facilitate the segmentation process, other systems use infrared cameras [50] or multi-camera systems [51]. These systems are more complex and more expensive to deploy than simple monocular visible-light camera systems.

In uncontrolled environments, recognizing human body postures in images and actions in videos becomes a challenging problem because of background clutter, the articulated structures of the human body, and the large variability of clothing. To overcome these difficulties, different methods based on directly matching *templates* to the targets have been studied. One method is to detect human body parts [74] [52] [53] and their spatial configuration in images. Body-part based methods usually only involve a few templates to represent each body part. The shortcoming of this method is that body parts are difficult to locate in many uncontrolled cases, mainly due to clothing changes, occlusion, and body-part deformation. Currently, body-part based schemes are used for recognizing relatively simple human postures such as walking [52] and running [54]. Another scheme recognizes human postures based on template matching. Most previous methods [9] [56] assume a relatively clean background. They rely on finding distinguishable features such as shape context [57] and do work well in uncluttered background settings. When background clutter increases, distinguished features are weakened and simple matching schemes cannot generate desirable results. More complex and powerful deformable template matching schemes need to be designed. The convex-programming based scheme we propose is a promising method to robustly and efficiently solve the problem [58] [59]. Based on the successive convex matching scheme, we set out methods to construct robust posture recognition systems by using appropriate local edge features and posture similarity measures. In experiments, we show successful application of the proposed scheme in detecting human postures in images. The proposed deformable matching scheme also acts as a robust front end for unsupervised action detection [78].

We further propose a successive convexification based video sequence matching method that allows us to detect actions in videos [79]. We represent an action as a sequence of body

postures with specific temporal constraints. We can then search for a given action by matching a sequence of coupled body posture templates to the video sequence. We formulate the matching problem as an energy minimization problem. The objective function is minimized such that the matching cost is low and at the same time we try to smooth the intra-frame matching and inter-frame object center's relative position. A center continuity constraint is important to force the matching to stick to one object in cluttered video where multiple objects may appear. As shown in our experiments, a greedy scheme such as ICM [3] is not robust enough if there is strong clutter or large deformation. Robust matching methods such as Graph Cut [7], Belief Propagation (BP) [5] and most recently a Linear Programming (LP) relaxation scheme [73] have been studied for finding correspondence in single image pairs using pairwise constraints. These methods are not easily extended to include the center continuity constraint. We consider a more straightforward approach — a successive convexification scheme to register template image sequences to targets in video. We extend the basic successive convexification method, reshaping the problem so that the inter-frame constraint can be introduced. Instead of directly solving the optimal labeling problem, the proposed scheme converts the optimization problem into easier convex problems and linear programming is applied to solve the sub-problems. An iterative process updates the trust region and successively improves the approximation. This scheme has many useful features: it involves only a small set of basis target points, and it is a strong approximation scheme. It is also found to be robust against strong clutter and large deformations, necessary for success of an action recognition scheme. After template to video registration, we compare the similarity of the matching targets in video with the templates by matching cost and degree of deformation.

1.6 Thesis Arrangement

We present the method of successive convexification in Chapter 2 in the context of object matching. In Chapter 3, large scale motion estimation is studied and we extend basic consistent labeling energy formulation to include occlusion inference. We show that the problem can still be solved with the convex relaxation method. Combining with a detail-preserving PDE scheme, a dense motion estimation method is studied. In Chapter 4, we study object tracking and propose an appearance adaptive object tracking method based on successive convexification. Mesh tracking is further combined with contour tracking for

better boundary extraction. In Chapter 5, we study the application of the convex relaxation scheme for human posture and action recognition. Successive convexification has also been used for deformable video sequence matching. We conclude the thesis in Chapter 6.

Chapter 2

Successive Convexification and Object Matching

In this chapter we study the method to convert a consistent labeling problem with convex regularization terms into a sequence of convex programming problems. The successive convexification scheme has useful properties which make it suitable for problems with very large label set. To facilitate the discussion, we study the proposed scheme in the context of object matching, which is a standard consistent labeling problem in computer vision. The method can be easily generalized into broader contexts.

2.1 Non-linear Optimization

The consistent labeling formulation of a matching problem can be stated as follows. We wish to assign a label $\mathbf{f_s}$ to each site $\mathbf{s}$ such that the objective function is minimized:

$$\min \left\{ \sum_{\mathbf{s} \in S} c(\mathbf{s}, \mathbf{f_s}) + \sum_{\{\mathbf{p},\mathbf{q}\} \in \mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} d(\mathbf{f_p} - \mathbf{p}, \mathbf{f_q} - \mathbf{q}) \right\}, \tag{2.1}$$

where $c(\mathbf{s}, \mathbf{f_s})$ is the cost of assigning a target point (label) $\mathbf{f_s}$ to a feature point (site) $\mathbf{s}$ on the template; $d(.)$ is a convex function and $d(\mathbf{f_p} - \mathbf{p}, \mathbf{f_q} - \mathbf{q})$ is to quantify the discrepancy of matching (label assignment) for neighboring sites $\mathbf{p}$ and $\mathbf{q}$ in S; S is a finite set of feature points on the template; $\mathcal{N}$ is the set of non-ordered neighboring site pairs. Here we define the pairs in $\mathcal{N}$ as those connected by edges in the Delaunay graph of point set S. $\mathbf{s}$ and $\mathbf{f_s}$ are both 2-D vectors in image matching. In this objective function, the first

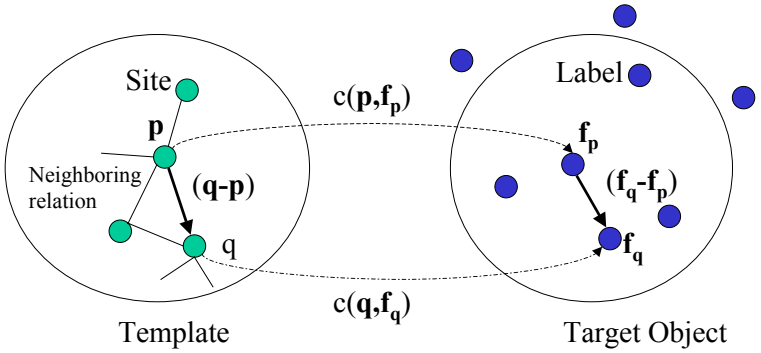


Figure 2.1: Consistent labeling for matching problem.

term is the label assignment cost; the second term is a regularization term to enforce the consistency of labeling for nearby sites. Nonnegative coefficients $\lambda_{\mathbf{p},\mathbf{q}}$ control the weight of the regularization term. Fig. 2.1 illustrates the labeling problem. In Fig. 2.1, points $\mathbf{p}$ and $\mathbf{q}$ are two neighboring sites and their targets are $\mathbf{f_p}$ and $\mathbf{f_q}$ respectively. Intuitively, we should minimize the matching costs and at the same time try to make the labeling consistent by minimizing the difference of vectors $\mathbf{q}-\mathbf{p}$ and $\mathbf{f_q}-\mathbf{f_p}$.

For labeling problems, the label set can be discrete or continuous. When the label set is discrete, we denote a problem as a discrete labeling problem, and otherwise as a continuous labeling problem. In discrete labeling problems, for each site $\mathbf{s}$ we can interpolate the labeling costs $c(\mathbf{s},\mathbf{t})$ piecewise-linearly over the $\mathbf{t}$ such that $c(\mathbf{s},\mathbf{t})$ become surfaces, and allow $\mathbf{f_s}$ to take on continuous values in the convex hull spanned by the discrete labels: we thus obtain the *continuous extension* of a discrete problem. Fig. 2.2 illustrates the concept of continuous extension for a 1-D problem. For 2-D labeling problems, $c(\mathbf{s},\mathbf{t})$ over $\mathbf{t}$ are surfaces. The piecewise-linear surface in 2-D is sometimes not unique if the surface is not convex. In such cases, the continuous extension surface is defined to be the one with the lowest value at each x-y coordinate $\mathbf{t}$.

Continuous labeling problems such as motion estimation or template matching can be well approximated by such a continuous extension of a discrete system. In the following discussions, without loss of generality we assume both the set of sites (template features points) S and set of labels (the target points) $\mathcal{L}_\mathbf{s}$ for each $\mathbf{s}\in S$ to be discrete.

In this thesis, we focus on the subset of consistent labeling problems in which $d(u,v)$ is a convex function. As will be shown later, the proposed scheme can be applied to problems with different convex regularization norms. To facilitate the discussion we study a special

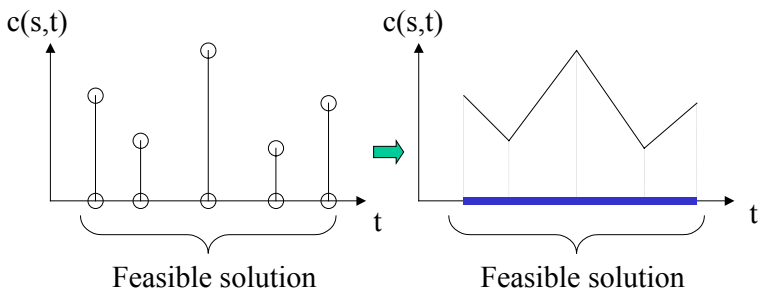


Figure 2.2: Continuous extension.

case where $d(\mathbf{u}, \mathbf{v}) = ||\mathbf{u}-\mathbf{v}||$, with $||\cdot||$ being the L_1 norm and then we extend the discussion to general problems. When the labeling problem degenerates into 1-D, a maximum-flow scheme [34] can be used to exactly solve the convex regularization term labeling problem. For problems with label space dimensionality greater than 1, the problem becomes much more complex. In the following, we assume labels are 2-D vectors. The methods proposed below can be easily extended to cases where the labels have higher dimensionality. To simplify notations, $c(\mathbf{s}, \mathbf{t})$ over $\mathbf{t}$ is also used to represent the continuous extension labeling cost surface for site $\mathbf{s}$.

2.2 Linear Programming Relaxation

The above energy optimization problem is nonlinear and usually non-convex, which makes it difficult to solve in this original form without a good initialization process. We now show how to approximate the problem by a linear programming formulation via linear approximation and variable relaxation when regularization terms use L_1 norms, as we outlined in [14] [60].

Basis labels and Objective Function Linearization A *"basis"* $\mathcal{B}_\mathbf{s}$ is selected for the labels for each site $\mathbf{s}$. Typically, for image matching problem, the basis is the set of target pixels at the lower convex hull vertices of the matching cost surfaces. Any 2-D target point $\mathbf{f_s}$ can certainly be represented as a linear combination of the basis, via $\mathbf{f_s} = \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\,\mathbf{j}$, where $\xi_{\mathbf{s},\mathbf{j}}$ are real-valued weighting coefficients. The labeling cost for $\mathbf{f_s}$ can then be approximated by the linear combination of the original, basis labeling costs $c(\mathbf{s}, \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\,\mathbf{j}) \approx \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\, c(\mathbf{s}, \mathbf{j})$. We also set constraints $\xi_{\mathbf{s},\mathbf{j}} \geq 0$ and $\sum_{\mathbf{j}\in\mathcal{B}_s} \xi_{\mathbf{s},\mathbf{j}} = 1$ for

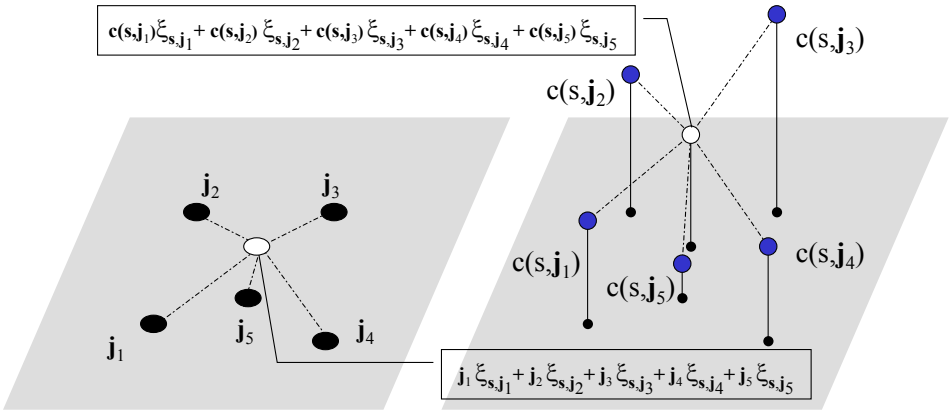


Figure 2.3: Basis and cost surface linearization.

each site $\mathbf{s}$. Clearly, if $\xi_{\mathbf{s},\mathbf{j}}$ are constrained to be 1 or 0, and the basis contains all the possible labels (target point candidates), i.e. $\mathcal{B}_\mathbf{s} = \mathcal{L}_\mathbf{s}$ (e.g., the whole target image), the above representation becomes exact. Fig. 2.3 illustrates the basis and cost surface linearization for a site $\mathbf{s}$. As will be shown later, the basis labels correspond to the x-y coordinates of the lower convex hull vertices of each labeling cost surface. At current stage, basis weighting coefficients ξ are not determined. They are determined as the linear program is optimized.

Note that $\mathbf{f}_\mathbf{s}$ are *not* constrained to the basis labels, but can be any convex combination of these labels. This is quite different from other discrete optimization methods such as BP. For BP, it is tricky when trying to reduce the target label set by discarding "unimportant" labels. If "good" labels happened to be discarded, BP solver will never retrieve them back and this results in gross errors. The proposed scheme has the good property to avoid such problems in the process of label set simplification.

To *linearize the regularization terms* in the nonlinear formulation (2.1) we can represent a variable whose absolute value appears in the objective function by the difference of two nonnegative auxiliary variables, and replace the absolute value term with the sum of the auxiliary variables in the objective function. If the problem is properly formulated, the sum will approach the absolute value of the variable when the linear program is optimized.

The Convex Relaxation and Properties Based on this linearization process, a linear programming relaxation of the problem can be stated in terms of the two vector components

$m = 1, 2$ (in place of x and y) as

$$\min \left\{ \sum_{\mathbf{s}\in S} \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c(\mathbf{s},\mathbf{j})\, \xi_{\mathbf{s},\mathbf{j}} \; + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{2} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}) \right\} \tag{2.2}$$

with constraints:

$$\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}} = 1 \;,\; \forall \mathbf{s} \in S$$

$$\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\, \phi_m(\mathbf{j}) = f_{\mathbf{s},m}, \;\; \forall \mathbf{s} \in S \,,\; \phi_m(\mathbf{j}) = (m\text{th component of } \mathbf{j}) \;\;, m = 1, 2$$

$$f_{\mathbf{p},m} - f_{\mathbf{q},m} - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q}) = f^+_{\mathbf{p},\mathbf{q},m} - f^-_{\mathbf{p},\mathbf{q},m} \,, \forall\, \{\mathbf{p},\mathbf{q}\} \in \mathcal{N} \,, m = 1, 2$$

$$\xi_{\mathbf{s},\mathbf{j}},\; f^+_{\mathbf{p},\mathbf{q},m},\; f^-_{\mathbf{p},\mathbf{q},m} \geq 0$$

The matching target point $\mathbf{f}(\mathbf{s}) = (f_{\mathbf{s},1}, f_{\mathbf{s},2})$. $f^+_{\mathbf{p},\mathbf{q},m}$ and $f^-_{\mathbf{p},\mathbf{q},m}$ are auxiliary variables.

In the linear program, one of the auxiliary variable pairs $f^+_{\mathbf{p},\mathbf{q},m}$, $f^-_{\mathbf{p},\mathbf{q},m}$ must be zero when the linear programming is optimized. Otherwise, we subtract the minimum of the value pair and get a new solution $f^+_{\mathbf{p},\mathbf{q},m} - \min(f^+_{\mathbf{p},\mathbf{q},m}, f^-_{\mathbf{p},\mathbf{q},m})$ and $f^-_{\mathbf{p},\mathbf{q},m} - \min(f^+_{\mathbf{p},\mathbf{q},m}, f^-_{\mathbf{p},\mathbf{q},m})$. Apparently, the new solution is still feasible and has at least one zero in these auxiliary variable pairs; the summation also becomes smaller. Thus, we have a better feasible solution and this contradicts the assumption that the linear program is optimized. Because one of $f^+_{\mathbf{p},\mathbf{q},m}$ and $f^-_{\mathbf{p},\mathbf{q},m}$ is zero, we must have $|f_{\mathbf{p},m} - f_{\mathbf{q},m} - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q})| = f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}$. Thus, the linear program is exactly equivalent to the general nonlinear formulation if the linearization assumption, $c(\mathbf{s}, \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\mathbf{j}) = \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}} c(\mathbf{s},\mathbf{j})$, holds. This would be true for problems with convex cost surfaces.

Property 2.1*: If* $\mathcal{B}_\mathbf{s} = \mathcal{L}_\mathbf{s}$, where $\mathcal{L}_\mathbf{s}$ is the entire label set of $\mathbf{s}$, and *the continuous extension cost function* $c(\mathbf{s},\mathbf{t})$ *is convex* over $\mathbf{t}$, $\forall \mathbf{s} \in S$ *, the LP exactly solves the continuous extension of the discrete labeling problem.*

Proof: We simply need to show that when the linear program is optimized, the minimizing LP configuration $\{\mathbf{f}^*_\mathbf{s} = \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi^*_{\mathbf{s},\mathbf{j}}\, \mathbf{j}\}$ also solves the continuous extension of the nonlinear problem. Since $c(\mathbf{s},\mathbf{t})$ is convex over $\mathbf{t}$, $\sum_{\mathbf{j}\in\mathcal{L}_\mathbf{s}} c(\mathbf{s},\mathbf{j})\xi^*_{\mathbf{s},\mathbf{j}} \geq c(\mathbf{s},\mathbf{f}^*_\mathbf{s})$. And based on the above analysis, when the LP is minimized we have $\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{2} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}) \equiv$

$\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} ||\mathbf{f}^*_{\mathbf{p}} - \mathbf{p} - \mathbf{f}^*_{\mathbf{q}} + \mathbf{q}||$, where $||\cdot||$ is the L_1 norm. Therefore

$$\begin{aligned} & \min\left\{\sum_{\mathbf{s}\in S,\mathbf{j}\in\mathcal{L}_s} c(\mathbf{s},\mathbf{j})\xi_{\mathbf{s},\mathbf{j}} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{2} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m})\right\} \\ \geq & \sum_{\mathbf{s}\in S} c(\mathbf{s},\mathbf{f}^*_{\mathbf{s}}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} ||\mathbf{f}^*_{\mathbf{p}} - \mathbf{p} - \mathbf{f}^*_{\mathbf{q}} + \mathbf{q}|| \end{aligned}$$

As well, we can always construct a feasible solution of LP that has no greater than objective function than that of the continuous extension of the nonlinear problem. Assuming solution $\mathbf{f}'_{\mathbf{s}}$ minimize the continuous extension of the non-linear problem, based on the definition, $\mathbf{f}'_{\mathbf{s}}$ must be located in the convex hull of $\mathcal{L}_s$. For each site $\mathbf{s}$, we find feasible ξ for the LP by solving the minimization problem

$$\min\left\{\sum_{,\mathbf{j}\in\mathcal{L}_s} c(\mathbf{s},\mathbf{j})\xi_{\mathbf{s},\mathbf{j}}\right\}$$

$$\begin{aligned} s.t. \quad \sum_{\mathbf{j}\in\mathcal{L}_s} \xi_{\mathbf{s},\mathbf{j}}\, \phi_m(\mathbf{j}) &= \phi_m(\mathbf{f}'_{\mathbf{s}}),\ m = 1,2 \\ \sum_{\mathbf{j}\in\mathcal{L}_s} \xi_{\mathbf{s},\mathbf{j}} &= 1\ ,\ \forall \mathbf{s}\in S \end{aligned}$$

We can set feasible values for $f^+_{\mathbf{p},\mathbf{q},m}$ and $f^-_{\mathbf{p},\mathbf{q},m}$: if $\phi_m(\mathbf{f}'_{\mathbf{p}}) - \phi_m(\mathbf{f}'_{\mathbf{q}}) - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q}) \geq 0$, $f^+_{\mathbf{p},\mathbf{q},m} = |\phi_m(\mathbf{f}'_{\mathbf{p}}) - \phi_m(\mathbf{f}'_{\mathbf{q}}) - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q})|$ and $f^-_{\mathbf{p},\mathbf{q},m} = 0$; else, $f^-_{\mathbf{p},\mathbf{q},m} = |\phi_m(\mathbf{f}'_{\mathbf{p}}) - \phi_m(\mathbf{f}'_{\mathbf{q}}) - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q})|$ and $f^+_{\mathbf{p},\mathbf{q},m} = 0$.

Based on the definition of continuous extension surface, we have $\min\left\{\sum_{\mathbf{j}\in\mathcal{L}_s} c(\mathbf{s},\mathbf{j})\xi_{\mathbf{s},\mathbf{j}}\right\} \leq c(\mathbf{s},\mathbf{f}'_{\mathbf{s}})$ for each $\mathbf{s}$ and therefore

$$\begin{aligned} & \min\left\{\sum_{\mathbf{s}\in S,\mathbf{j}\in\mathcal{L}_s} c(\mathbf{s},\mathbf{j})\xi_{\mathbf{s},\mathbf{j}} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{2} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m})\right\} \\ \leq & \sum_{\mathbf{s}\in S} c(\mathbf{s},\mathbf{f}'_{\mathbf{s}}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} ||\mathbf{f}'_{\mathbf{p}} - \mathbf{p} - \mathbf{f}'_{\mathbf{q}} + \mathbf{q}|| \end{aligned}$$

Therefore $\mathbf{f}^*_{\mathbf{s}}$ optimizes the continuous extension of the nonlinear problem:

$$\sum_{\mathbf{s}\in S} c(\mathbf{s},\mathbf{f}^*_{\mathbf{s}}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} ||\mathbf{f}^*_{\mathbf{p}} - \mathbf{p} - \mathbf{f}^*_{\mathbf{q}} + \mathbf{q}|| = \sum_{\mathbf{s}\in S} c(\mathbf{s},\mathbf{f}'_{\mathbf{s}}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} ||\mathbf{f}'_{\mathbf{p}} - \mathbf{p} - \mathbf{f}'_{\mathbf{q}} + \mathbf{q}||$$

The property follows.

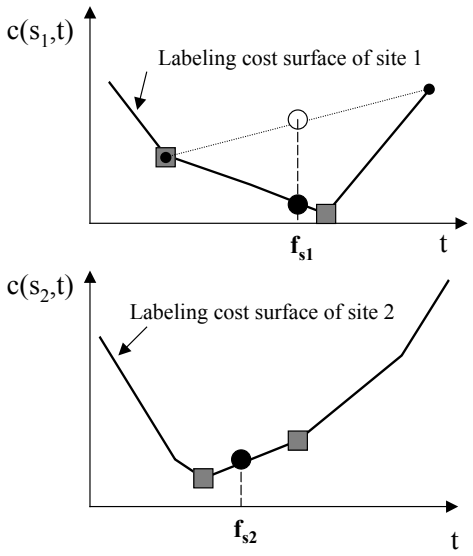


Figure 2.4: Optimization for convex labeling problems.

Fig. 2.4 illustrates the property of linear programming relaxation for convex labeling problems. We assume there are only two neighboring sites in this example. When the linear program is optimized, it must set the coefficients ξ to make the linear combination of costs fall onto the convex cost surfaces. The linear program compromises the labeling cost and labeling discrepancy to make the whole energy minimized. Since the objective function is non-differentiable, a naive gradient descent method cannot be directly applied. Convex programming is a natural choice for such a problem. For this 1-D problem, there are in fact at most 2 non-zero ξ for each site and the corresponding labels must be adjacent. The vertices with non-zero ξ are shown as the gray rectangles. Other configurations will result in higher objective functions as illustrated in Fig. 2.4. We will show that this observation is in fact valid for 2-D and higher dimensional problems, for which the optimal configuration of ξ has only a small number of non-zeros.

In practice, the cost function $c(\mathbf{s}, \mathbf{t})$ is usually highly non-convex over $\mathbf{t}$ for each site $\mathbf{s}$. In this situation, the linear program approximates the original non-convex problem:

Property 2.2*: The linear program solves the continuous extension of the reformulated discrete labeling problem, with $c(\mathbf{s}, \mathbf{t})$ replaced by its the lower convex hull surface $\breve{c}(\mathbf{s}, \mathbf{t})$ for each site $\mathbf{s}$. $\breve{c}(\mathbf{s}, \mathbf{t})$ is the lower convex hull surface of 3-D points $(\phi_1(\mathbf{j}), \phi_2(\mathbf{j}), c(\mathbf{s}, \mathbf{j})), \mathbf{j} \in \mathcal{L}_\mathbf{s}$.*

The proof is similar to that of Property 1, by replacing $c(\mathbf{s}, \mathbf{t})$ in the nonlinear function

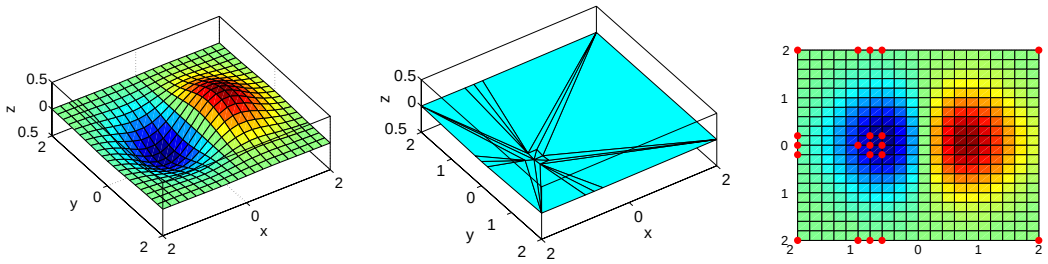


Figure 2.5: Lower convex hull. Left: a cost surface; Middle: Lower convex hull facets; Right: The label basis $\mathcal{B}_{\mathbf{s}}$ coordinates of the lower convex hull vertices (Solid dots are basis points).

with its lower convex hull. An example for the lower convex hull and the coordinates of the lower convex hull vertices is illustrated in Fig. 2.5. As shown in Fig. 2.5, a non-convex cost surface is converted to a simple lower convex hull, which can be represented with a small number of vertices. For matching applications, the surface is the matching cost surface. Note that in general the surface may have holes, or consist only of irregular discrete 3-D points in the label vs. cost space, e.g. if we only select edge points in the target images for matching. Each site $\mathbf{s}$ is associated with its own energy surface for the first term in (2.1).

Property 2.3*: For each site* $\mathbf{s} \in S$*, we need only consider the basis set* $\mathcal{B}_{\mathbf{s}}$ *comprised of the vertex coordinates of the lower convex hull of* $(\phi_1(\mathbf{j}), \phi_2(\mathbf{j}), c(\mathbf{s}, \mathbf{j}))$, $\mathbf{j} \in \mathcal{L}_{\mathbf{s}}$.

Proof: Removing some variables $\xi_{\mathbf{s},\mathbf{j}}$ from LP is equivalent to setting these variables to zero in the constraints. Since more constraints are included, the linear program involving only $\xi_{\mathbf{s},\mathbf{j}}$ corresponding to the lower convex hull vertices must have no smaller objective function when optimized. On the other hand, based on the definition of lower convex hull, for $\mathbf{f}_{\mathbf{s}}$ that minimize the original LP relaxation, we can find a feasible solution of the simplified linear program that has an optimum not greater than that of the original LP. Therefore, these two linear programs must be equivalent. The property follows.

Thus, we can use only the smallest basis set — there is no need to include all the labeling assignment costs in the optimization. This is one of the key steps to speed up the algorithm. As shown in Fig. 2.5, basis labels usually have small number. The following example shows how a labeling problem can be constructed based only on the basis labels.

Example 2.1: Assume that we have a linear programming relaxation for a 1-D labeling

problem as

$$\begin{aligned} & \min\{\sum_{j=1}^{10} c(1,j)\xi_{1,j} + \sum_{j=1}^{10} c(2,j)\xi_{2,j} + 0.5(f_{1,2}^{+} + f_{1,2}^{-})\} \\ s.t. \quad & \sum_{j=1}^{10} \xi_{i,j} = 1, \quad i = 1,2 \\ & \sum_{j=1}^{10} j\xi_{i,j} = f_i, \quad i = 1,2 \\ & f_1 - f_2 = f_{1,2}^{+} - f_{1,2}^{-} \end{aligned}$$

where $c(1,j)$ ={1.5, 4, 5, 5, 6, 1.7, 4, 5, 2, 2}, $c(2,j)$ ={5, 5, 5, 1, 1, 3, 4, 1, 2, 5}; all the variables are greater than or equal to 0. This is the LP in example 2.2 with the largest trust regions for each site. This LP can be simplified as

$$\begin{aligned} & \min\left\{\sum_{j=1,6,10} c(1,j)\xi_{1,j} + \sum_{j=1,4,8,9,10} c(1,j)\xi_{2,j} + 0.5(f_{1,2}^{+} + f_{1,2}^{-})\right\} \\ s.t. \quad & \sum_{j=1,6,10} \xi_{1,j} = 1, \quad \sum_{j=1,4,8,9,10} \xi_{2,j} = 1 \\ & \sum_{j=1,6,10} j\xi_{1,j} = f_1, \quad \sum_{j=1,4,8,9,10} j\xi_{2,j} = f_2 \\ & f_1 - f_2 = f_{1,2}^{+} - f_{1,2}^{-} \end{aligned}$$

and both linear programs have the solution of $\xi_{1,1} = 0.4$, $\xi_{1,6} = 0.6$, $\xi_{2,4} = 1$, other ξ are 0 and $f_1 = 4, f_2 = 4$. In this example, as illustrated in Fig. 2.6, for site 1, labels {1, 6, 10} are basis labels; {(1,1.5), (6,1.7), (10,2)} are vertices of the lower convex hull; labels {1, 6} have non-zero weights. For site 2, {1, 4, 8, 9, 10} are basis labels; {(1,5), (4,1), (8,1), (9,2), (10,5)} are lower convex hull vertices; label 4 has non-zero weight. This example is for 1-D problem. It equally applies to higher dimensional problems.

After the convexification process, the original non-convex optimization problem turns into a convex problem and an efficient linear programming method can be used to yield a global optimal solution for the approximated problem. It should be noted that, although this is a convex problem, standard local optimization schemes are found to work poorly because of quantization noise and large flat areas in the convexified objective function.

Approximating the matching cost by its lower convex hull is intuitively attractive since in the ideal case when model (2.1) holds exactly, the true matching will have the lowest

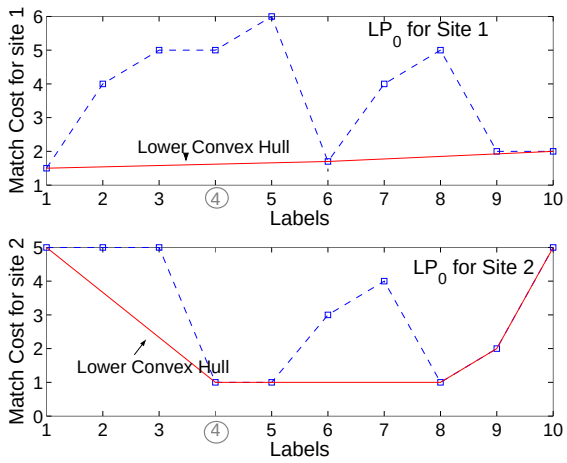


Figure 2.6: Basis labels and lower convex hull. Labels in circles are LP continuous solutions.

cost and the convexified optimum will be the exact optimum. As will be shown later in an analysis of complexity, this scheme also decouples the size of target label set and the size of the convex programming problems.

The solution of the relaxation scheme also has the following structure property.

Property 2.4*: If the lower convex hull* of *the cost function* $c(\mathbf{s}, \mathbf{t})$ is strictly convex over $\mathbf{t}$ for each $\mathbf{s}$, *its non-zero basis labels must be "adjacent".*

Proof: Here "adjacent" means the convex hull of the nonzero-weighted basis labels cannot contain other basis labels. Assume this does not hold for site $\mathbf{s}$, and the nonzero-weight basis labels are $\mathbf{j}_k$, $k = 1..K$. Then, there exists a basis label $\mathbf{j}_r$ located inside the convex hull of $\mathbf{j}_k$, $k = 1..K$. Thus, $\exists\, \alpha_k$ such that $\mathbf{j}_r = \sum_{k=1}^{K} \alpha_k \mathbf{j}_k$ with $\sum_{k=1}^{K} \alpha_k$=1, $\alpha_k \geq 0$. According to the *Karush-Kuhn-Tucker Condition* $(KKTC)$, there exist $\lambda_1, \lambda_2, \lambda_3$ and $\mu_{\mathbf{j}}$ such that

$c(\mathbf{s}, \mathbf{j}) + \lambda_1 + \lambda_2 \phi_1(\mathbf{j}) + \lambda_3 \phi_2(\mathbf{j}) - \mu_{\mathbf{j}} = 0$ and

$\xi_{\mathbf{s},\mathbf{j}} \cdot \mu_{\mathbf{j}} = 0, \mu_{\mathbf{j}} \geq 0, \forall \mathbf{j} \in \mathcal{B}_{\mathbf{s}}$.

This results from the Euler equation with respect to $\xi_{\mathbf{s},\mathbf{j}}$ in (2.2), taking into account the constraints. Therefore we have,

$c(\mathbf{s}, \mathbf{j}_k) + \lambda_1 + \lambda_2 \phi_1(\mathbf{j}_k) + \lambda_3 \phi_2(\mathbf{j}_k) = 0, k = 1..K$

$c(\mathbf{s}, \mathbf{j}_r) + \lambda_1 + \lambda_2 \phi_1(\mathbf{j}_r) + \lambda_3 \phi_2(\mathbf{j}_r) \geq 0$

On the other hand,

$c(\mathbf{s}, \mathbf{j}_r) + \lambda_1 + \lambda_2 \phi_1(\mathbf{j}_r) + \lambda_3 \phi_2(\mathbf{j}_r)$

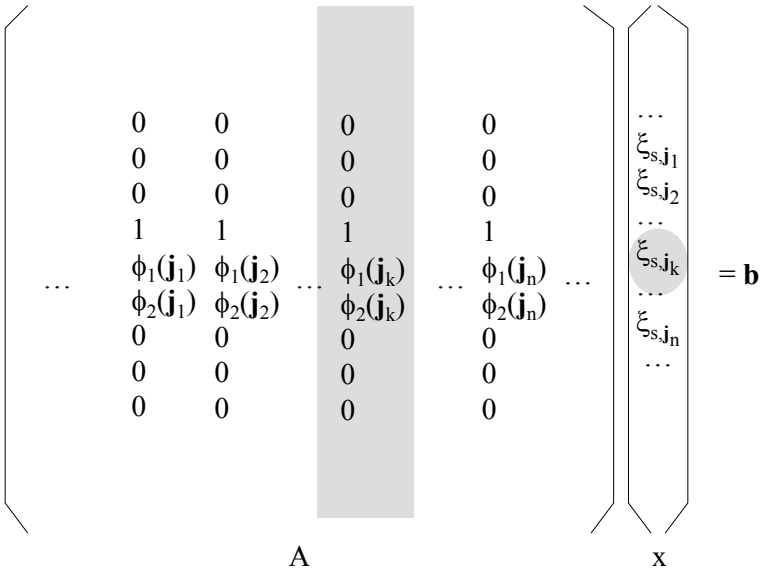


Figure 2.7: Each variable ξ introduces a column in constraint matrix.

$= c(\mathbf{s}, \sum_{k=1}^{K} \alpha_k \mathbf{j}_k) + \lambda_1 + \lambda_2 \phi_1(\sum_{k=1}^{K} \alpha_k \mathbf{j}_k) + \lambda_3 \phi_2(\sum_{k=1}^{K} \alpha_k \mathbf{j}_k)$
$< \sum_{k=1}^{K} \alpha_k c(\mathbf{s}, \mathbf{j}_k) + \lambda_1 + \lambda_2 \sum_{k=1}^{K} \alpha_k \phi_1(\mathbf{j}_k) + \lambda_3 \sum_{k=1}^{K} \alpha_k \phi_2(\mathbf{j}_k) = 0$
which contradicts *KKTC*. The property follows.

If we use Dantzig's simplex method [28] to solve the LP, we have the following property:

Property 2.5*: For 2-D labeling problems, using the simplex method, there will be at most 3 nonzero-weight basis labels for each site.*

Proof: This property is due to the linear programming property: if the optimum of an LP exists, the optimum must be located at one of the "extreme" points of the feasible region [41]. The extreme points of LP correspond to basic feasible solutions. We denote the constraints of our linear program by $A\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$. Each basic feasible solution of LP has the format $[B^{-1}b, 0]^T$ where B is an invertible matrix composed of the columns of matrix A corresponding to the basic variables. For site $\mathbf{s}$, variable $\xi_{\mathbf{s},\mathbf{j}}$ introduces a column $[0, .., 0, 1, \phi_1(\mathbf{j}), \phi_2(\mathbf{j}), 0, ..., 0]^T$ in A, as illustrated in Fig. 2.7. It is not difficult to show that the sub-matrix generated by these columns for a single site has a rank at most 3. Therefore, we can have at most three ξ for each site in the basic variable set. This implies that the optimum solution has at most three nonzero ξ for each site.

Importantly, Property 2.4 implies that for each site the proposed LP relaxation **searches only the triangles consisting of the lower convex hull vertices**, in an efficient energy

descent manner. And note that the triangles may degenerate into lines or points. During each iteration, the simplex scheme chooses one vertex label to remove from the basis and a new one to enter the basis. The number of these lower convex hull vertex labels is usually several orders of magnitude smaller than the number of candidate labels for each site. Hence this method greatly improves efficiency of the searching process. Note that basic variables here define the extreme points in the feasible region of the linear program, which is different from basis labels that correspond to the lower convex hull vertices of matching cost surfaces.

For the simplex method, the initial basic variables are set as follows:

- Only one randomly selected $\xi_{\mathbf{s},\mathbf{j}}$ is selected as basic LP variable for each site $\mathbf{s}$.
- $f_{\mathbf{s},m}, m = 1..2$, are basic LP variables for each site $\mathbf{s}$.
- Based on the above basic label selection we can easily get $f_{\mathbf{s},m}$ since only one $\xi_{\mathbf{s},\mathbf{j}}$ is 1 for each site $\mathbf{s}$. For $m = 1..2$, if $f_{\mathbf{p},m} - \phi_m(\mathbf{p}) - f_{\mathbf{q},m} + \phi_m(\mathbf{q}) \geq 0$ and $\{\mathbf{p}, \mathbf{q}\} \in \mathcal{N}$, $f^+_{\mathbf{p},\mathbf{q},m}$ is basic; else $f^-_{\mathbf{p},\mathbf{q},m}$ is set to be basic.

The above basic label selection method in fact corresponds to randomly choosing initial labels for each site. Since the linear program can be globally optimized, its solution is not dependent on the initial starting point. We can also select the label with the lowest labeling cost for each site and set the corresponding $\xi_{\mathbf{s},\mathbf{j}}$ to be basic variable; such an initialization process often speeds up the convergence.

Property 2.6: *When the simplex method optimizes the linear program, the lower convex hull vertices corresponding to non-zero weight basis labels for each site are located in the same planar patch.*

Proof: Since we have at most three non-zero weight labels, the vertices corresponding to these labels must span a triangle patch (or its degenerate, a point and a line segment). The line segment or the planar patch must be an edge or be within a planar patch of the lower convex hull; otherwise, the line segment and the triangle patch must be above the lower convex hull and we can construct another LP solution that has smaller objective function which contradicts the assumption. The property follows.

Example 2.2 (Matching Triangles): Fig. 2.8 illustrates the solution procedure of the simplex method for an image matching problem. In this simple example, three feature points are selected on the toy object and form a triangular graph template. All the dark pixels in the target image are matching candidates for each of the feature points. Figs. 2.8 (c, d, e)

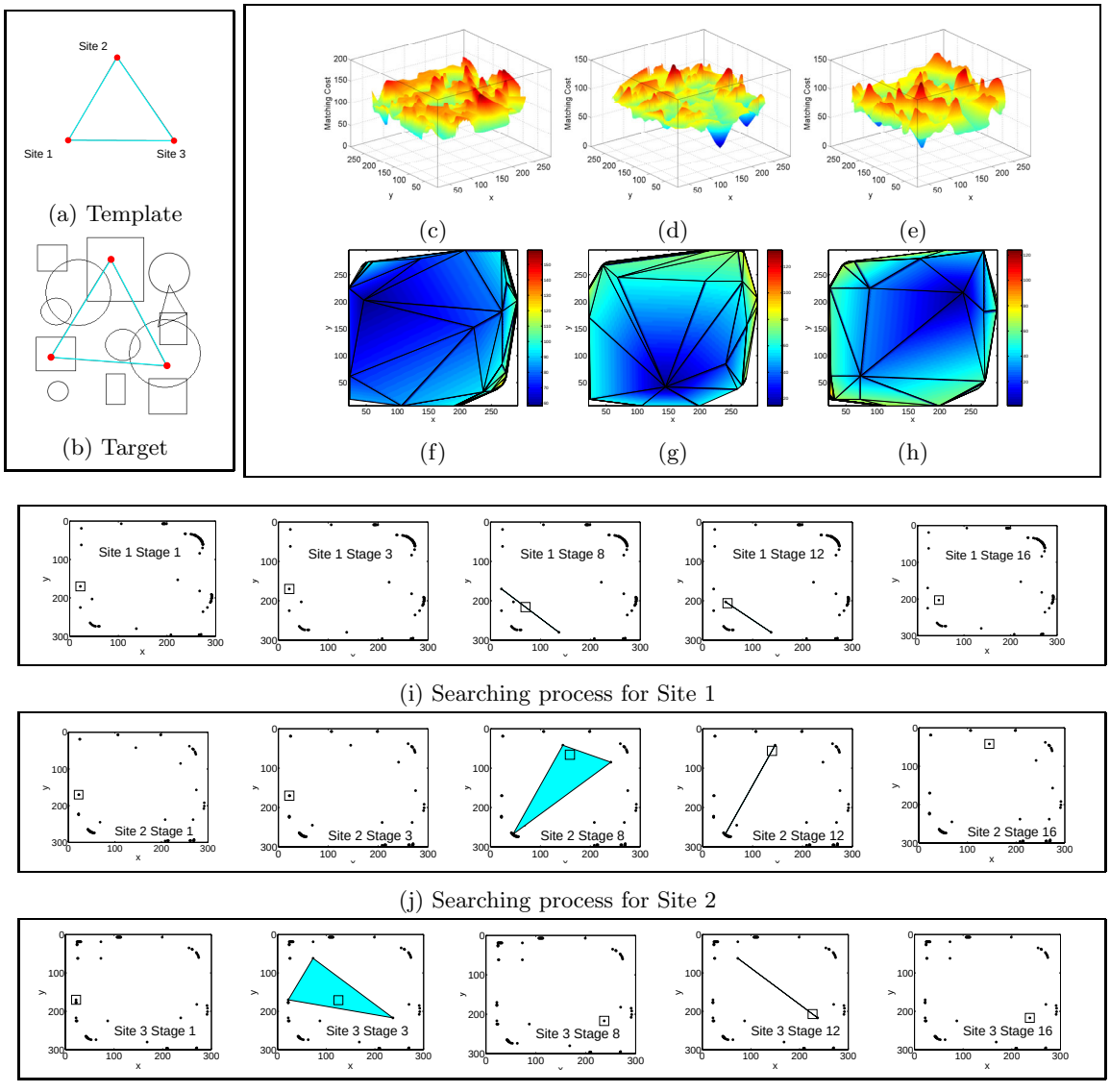
show the matching cost surfaces for each of the three points on the template, based on the normalized mean absolute difference of 7 ×7 image blocks of the distance transformation of the template and target images. Such a measure is used in this chapter for binary object matching. More details on the feature are presented in Section 2.10. Figs. 2.8 (f, g, h) are the lower convex hull surfaces for the respective cost surfaces for each of the three sites. We formulate the linear program based on the proposed scheme and we use the simplex method to solve the LP. Figs. 2.8 (i, j, k) show the basic label updating process. The black dots indicate the target points located at the coordinates of the lower convex hull vertices. The target points corresponding to the basic variables are connected by lines. The small rectangle is the weighted linear combination of the target points corresponding to the basic labels at each stage. As expected, the simplex method for the proposed LP only checks triangles or their degenerates formed by basis target points. In each iteration, only one vertex of the triangle may change. When the search terminates, the patch generated by the basic variables for each site must correspond to one of the facets (edges or vertices) of the lower convex hull for each site.

2.3 Approximation Properties of Single Relaxation

Although the continuous-weight solution obtained by LP is itself a good solution to the original problem for many applications, e.g. estimating medium scale motion [14], in many cases we would like to obtain the discrete solution in which the matching target points are only extracted from the original, discrete, target point candidate set. In the following discussions, we present bounds for the approximation scheme when the continuous solution of the linear programming is converted to a discrete one. These "rounding" processes are also useful in estimating upper bounds and locating *anchors* (see below) in the successive convexification process.

If the label costs are not bounded above, there is no upper bound for the energy of direct LP solution when applied to the continuous extension nonlinear problem. But we can take a different approach. When converting a continuous solution to a solution feasible in the discrete domain, we enforce that $\xi_{\mathbf{s},\mathbf{j}}$ only have one single 1 for each site. There are two methods to enforce the constraint. The first one converts the largest $\xi_{\mathbf{s},\mathbf{j}}$ for each site to 1 and others to 0. The following proposition gives an upper bound for the rounding process.

Proposition 2.1: *For each* $\mathbf{s} \in S$*, if we round the largest* $\xi_{\mathbf{s},\mathbf{j}}$ *to 1 and the rest to zero,*



(a) Template

(b) Target

(c) (d) (e)

(f) (g) (h)

(i) Searching process for Site 1

(j) Searching process for Site 2

(k) Searching process for Site 3

Figure 2.8: Example of Matching. (a): Template image; (b): Target image and LP matching result; (c, d, e): Matching cost surface for Sites 1, 2 and 3; (f, g, h): Lower convex hull of the matching cost surface for Sites 1, 2, and 3; (i, j, k): Triangular basis updating for Sites 1, 2, and 3 (noting that triangles often degenerate).

thus producing a rounded version $\hat{\mathbf{f}}_\mathbf{s}$ *of the continuous solution* $\mathbf{f}_\mathbf{s}$*, we have an approximation algorithm bounded above by* $3E_{opt}+\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}(||\hat{\mathbf{f}}_\mathbf{p}-\mathbf{f}_\mathbf{p}||+||\hat{\mathbf{f}}_\mathbf{q}-\mathbf{f}_\mathbf{q}||)$ *. E_{opt} is the energy of the optimum solution.*

Proof: Firstly, $\sum_\mathbf{s} c(\mathbf{s},\hat{\mathbf{f}}_\mathbf{s})+\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\hat{\mathbf{f}}_\mathbf{p}-\mathbf{p}-\hat{\mathbf{f}}_\mathbf{q}+\mathbf{q}|| \le \sum_\mathbf{s} c(\mathbf{s},\hat{\mathbf{f}}_\mathbf{s})+\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}$ $(||\hat{\mathbf{f}}_\mathbf{p}-\mathbf{f}_\mathbf{p}||+||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}||+||\mathbf{f}_\mathbf{q}-\hat{\mathbf{f}}_\mathbf{q}||)$ by the triangle inequality. A direct conclusion from Property 2.5 is that for each site $\mathbf{s}$, $\max_\mathbf{j}(\xi_{\mathbf{s},\mathbf{j}}) \ge 1/3$ (else the sum would be necessarily less than 1). Therefore, rounding ξ to $\hat{\xi}$, $\sum_\mathbf{s} c(\mathbf{s},\hat{\mathbf{f}}_\mathbf{s}) \to \sum_{\mathbf{s}\in S}\ \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c(\mathbf{s},\mathbf{j})\,\hat{\xi}_{\mathbf{s},\mathbf{j}} \le 3\sum_{\mathbf{s}\in S}\ \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c(\mathbf{s},\mathbf{j})\,\xi_{\mathbf{s},\mathbf{j}}$ (the maximum increase is times 3). Considering the fact that $E_{LP} = \sum_{\mathbf{s}\in S}\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c(\mathbf{s},\mathbf{j})\,\xi_{\mathbf{s},\mathbf{j}} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}|| \le E_{opt}$, the proposition follows. For practical computer vision problems, most of the $\xi_{\mathbf{s},\mathbf{j}}$ in fact approach 1 or 0 and therefore $\hat{\mathbf{f}}_\mathbf{s}$ approaches $\mathbf{f}_\mathbf{s}$. Thus the simple rounding scheme approaches a constant approximation method.

One of the problems of the above rounding process is that it does not consider the neighbor's status and may result in topology changes. To solve this problem, we propose another method in the rounding process: we check the discrete target points and select the one that minimizes the nonlinear objective function, given the configuration of continuous labeling results defined by the LP solution. This step is similar to a single iteration of an ICM algorithm with LP solution as the initial value. We call the new rounding scheme a *consistent rounding* process.

Proposition 2.2: *The energy with consistent rounding is bounded above by* $3E_{opt} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{m}_\mathbf{p}-\mathbf{f}_\mathbf{p}|| + ||\mathbf{m}_\mathbf{q}-\mathbf{f}_\mathbf{q}||)$*, where E_{opt} is the energy of the optimum solution*; $\mathbf{m}_\mathbf{s}$ is the global optimal solution; $\mathbf{f}_\mathbf{s}$ is the continuous labeling solution of LP.

Proof: The proof is simple but lengthy. We denote $\mathbf{r}_\mathbf{s}$ as the consistent rounding solution.

$\sum_\mathbf{s} c(\mathbf{s},\mathbf{r}_\mathbf{s}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{r}_\mathbf{p}-\mathbf{p}-\mathbf{r}_\mathbf{q}+\mathbf{q}||$

$\le \sum_\mathbf{s} c(\mathbf{s},\mathbf{r}_\mathbf{s}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{r}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}|| + ||\mathbf{r}_\mathbf{q}-\mathbf{q}-\mathbf{f}_\mathbf{p}+\mathbf{p}|| + ||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}||)$

$= \sum_\mathbf{s} c(\mathbf{s},\mathbf{r}_\mathbf{s}) + \sum_\mathbf{s}\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{s})}\lambda_{\mathbf{s},\mathbf{p}}||\mathbf{r}_\mathbf{s}-\mathbf{s}-\mathbf{f}_\mathbf{p}+\mathbf{p}|| + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}||$

Recalling the rule of consistent rounding,

$c(\mathbf{s},\mathbf{r}_\mathbf{s}) + \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{s})}\lambda_{\mathbf{s},\mathbf{p}}||\mathbf{r}_\mathbf{s}-\mathbf{s}-\mathbf{f}_\mathbf{p}+\mathbf{p}|| \le c(\mathbf{s},\mathbf{m}_\mathbf{s}) + \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{s})}\lambda_{\mathbf{s},\mathbf{p}}||\mathbf{m}_\mathbf{s}-\mathbf{s}-\mathbf{f}_\mathbf{p}+\mathbf{p}||$

Therefore

$\sum_\mathbf{s} c(\mathbf{s},\mathbf{r}_\mathbf{s}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{r}_\mathbf{p}-\mathbf{p}-\mathbf{r}_\mathbf{q}+\mathbf{q}||$

$\le \sum_\mathbf{s} c(\mathbf{s},\mathbf{m}_\mathbf{s}) + \sum_\mathbf{s}\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{s})}\lambda_{\mathbf{s},\mathbf{p}}||\mathbf{m}_\mathbf{s}-\mathbf{s}-\mathbf{f}_\mathbf{p}+\mathbf{p}|| + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}||$

$= \sum_\mathbf{s} c(\mathbf{s},\mathbf{m}_\mathbf{s}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{m}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}|| + ||\mathbf{m}_\mathbf{q}-\mathbf{q}-\mathbf{f}_\mathbf{p}+\mathbf{p}||) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f}_\mathbf{p}-\mathbf{p}-\mathbf{f}_\mathbf{q}+\mathbf{q}||$

$\leq \sum_{\mathbf{s}} c(\mathbf{s}, \mathbf{m_s}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{m_p} - \mathbf{p} - \mathbf{m_q} + \mathbf{q}|| + ||\mathbf{f_q} - \mathbf{m_q}||$
$+||\mathbf{m_q} - \mathbf{q} - \mathbf{m_p} + \mathbf{p}|| + ||\mathbf{m_p} - \mathbf{f_p}||) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f_p} - \mathbf{p} - \mathbf{f_q} + \mathbf{q}||$
$\leq \sum_{\mathbf{s}} c(\mathbf{s}, \mathbf{m_s}) + 2\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\mathbf{m_p} - \mathbf{p} - \mathbf{m_q} + \mathbf{q}|| + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f_p} - \mathbf{p} - \mathbf{f_q} + \mathbf{q}||$
$+\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{f_q} - \mathbf{m_q}|| + ||\mathbf{m_p} - \mathbf{f_p}||)$

Noticing that $\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f_p} - \mathbf{p} - \mathbf{f_q} + \mathbf{q}|| \leq E_{opt}$, the proof is complete.

Consistent rounding behaves better in preserving the adjacency of labels while we shrink the trust region. As discussed above, a single LP relaxation approximates the original problem's matching cost functions by their lower convex hulls. In real applications, several target points may have equal matching cost and, even worse, some incorrect matches may have lower cost. In this case, because of the convexification process, many local structures are removed, which on the one hand facilitates the search process by removing many false local minimums and on the other hand makes the solution not exactly locate on the true global minimum. In the next section, we set out a successive convexification scheme to solve the problem.

2.4 Successive Convexification

Here we propose a successive relaxation method to solve the nonlinear optimization problem by constructing linear programming recursively based on the previous searching result and gradually shrinking the matching trust region for each site systematically. Assume $\mathcal{B}_{\mathbf{s}}^n$ is the basis label set for site $\mathbf{s}$ at stage n linear programming. The trust region $\mathcal{U}_{\mathbf{s}}^n$ of site $\mathbf{s}$ is determined by the previous relaxation solution, and a trust region diameter d_n. We define $\mathcal{Q}_{\mathbf{s}}^n = \mathcal{L}_{\mathbf{s}} \cap \mathcal{U}_{\mathbf{s}}^n$: the reduced target point space falling within the current diameter. The nth basis $\mathcal{B}_{\mathbf{s}}^n$ is specified by $\mathcal{B}_{\mathbf{s}}^n = \{$the vertex coordinates of the lower convex hull of $\{(\phi_1(\mathbf{j}), \phi_2(\mathbf{j}), c(\mathbf{s},\mathbf{j})), \forall \mathbf{j} \in \mathcal{Q}_{\mathbf{s}}^n\}\}$, where $c(\mathbf{s},\mathbf{j})$ is the cost of assigning label $\mathbf{j}$ to site $\mathbf{s}$.

Algorithm 2.1. *Successive Convexification*

1. *Set* $n = 0$*; Set initial (maximum) diameter* $= d_0$;
2. *FOREACH*$(\mathbf{s} \in S)$
3. *Calculate the cost function* $\{c(\mathbf{s},\mathbf{j}), \forall \mathbf{j} \in \mathcal{Q}_{\mathbf{s}}^0\}$*;*
4. *Convexify* $\{(\phi_1(\mathbf{j}), \phi_2(\mathbf{j}), c(\mathbf{s},\mathbf{j}))\}, \mathbf{j} \in \mathcal{Q}_{\mathbf{s}}^{\mathbf{0}}$ *and find basis* $\mathcal{B}_{\mathbf{s}}^0$*;*
5. *Construct and solve* $\mathcal{LP}_0$;
6. *WHILE* $(d_n \geq d_{\min})$

7. $n \leftarrow n+1$;
8. $d_n = d_{n-1} - \delta_n$;
9. *FOREACH*$(\mathbf{s} \in S)$
10. *IF* ($\mathcal{Q}_{\mathbf{s}}^n$ *is empty*) $\mathcal{Q}_{\mathbf{s}}^n = \mathcal{Q}_{\mathbf{s}}^{n-1}$; $\mathcal{U}_{\mathbf{s}}^n = \mathcal{U}_{\mathbf{s}}^{n-1}$;
11. *ELSE update* $\mathcal{U}_{\mathbf{s}}^n$, $\mathcal{Q}_{\mathbf{s}}^n$;
12. *Re-convexify* $\{(\phi_1(\mathbf{j}), \phi_2(\mathbf{j}), c(\mathbf{s}, \mathbf{j}))\}, \mathbf{j} \in \mathcal{Q}_{\mathbf{s}}^{\mathbf{n}}$ *and redetermine basis* $\mathcal{B}_{\mathbf{s}}^n$;
13. *Construct and solve* $\mathcal{LP}_n$;
14. *Output* $\mathbf{f}_{\mathbf{s}}^*, \forall \mathbf{s} \in S$.

We use an *anchor* to control the trust region of each site in the iteration. For image matching, the trust region for each site is a rectangular area in the target image. For general labeling problems, the trust region is a "cube" in the label space. We keep the anchor in the new trust region for each site and shrink the boundaries inwards. If the anchor is on a boundary of the previous trust region, other boundaries are moved inwards. We also require that new anchors have energy not greater than the previous estimation: the anchors are updated only if new ones have smaller energy. A simple scheme is to select anchors as the solution of the previous LP. Unfortunately, in the worst case this simple scheme has solutions whose objective function is arbitrarily far from the optimum. In fact, the continuous solution could be far away from the discrete label site. So instead we use the consistent rounding solution as presented in the last section for obtaining the anchor. It is not difficult to verify that the necessary condition for successive relaxation to converge to the global minimum is that $\mathcal{LP}_n \leq E_{opt}$, where E_{opt} is the global minimum of the nonlinear problem. Since the global minimum of the function is unknown, we estimate an upper bound $\mathcal{E}^+$of E_{opt} in the current iteration. The upper bound $\mathcal{E}^+$ is then set to be the objective function of anchors. The objective function for $\mathcal{LP}_n$ must be less than or equal to $\mathcal{E}^+$. This iterative procedure guarantees that the objective function of the proposed multi-step scheme is at least as good as a single relaxation scheme. In the following example, we use a simple 1-D labeling problem to illustrate the solution procedure.

Example 2.3 (A 1-D problem): Assume there are two sites $\{1, 2\}$ and for each site the label set is $\{1..10\}$. The objective function is $\min_{\{f_1,f_2\}} \{c(1, f_1) + c(2, f_2) + \lambda|f_1 - f_2|\}$. In this example we assume that $c(1, j)$=$\{1.5,\ 4,\ 5,\ 5,\ 6,\ 1.7,\ 4,\ 5,\ 2,\ 2\ \}$; $c(2, j)$=$\{5,\ 5,\ 5,\ 1,\ 1,\ 3,\ 4,\ 1,\ 2,\ 5\}$, and $\lambda = 0.5$. Based on the proposed scheme, the problem is solved by the 4-step LPs: $\mathcal{LP}_0$, $\mathcal{LP}_1$, $\mathcal{LP}_2$, $\mathcal{LP}_3$.

- In $\mathcal{LP}_0$ the trust regions for sites 1 and 2 both start as the whole label space $[1, 10]$. Constructing $\mathcal{LP}_0$ based on the proposed scheme corresponds to solving an approximated problem in which c for site 1 and 2 are replaced by their lower convex hulls respectively (see Fig. 2.9). Step $\mathcal{LP}_0$ uses convex hull basis labels {1, 6, 10} for site 1 and {1, 4, 8, 9, 10} for site 2. $\mathcal{LP}_0$ finds a solution with nonzero weights $\xi_{1,1} = 0.4$ and $\xi_{1,6} = 0.6$, $f_1 = 0.4 * 1 + 0.6 * 6 = 4$; and $\xi_{2,4} = 1$, and resulting continuous label LP solution $f_2 = 4$. Based on the proposed rules for anchor selection, we fix site 1 at label 4 and search for the best anchor for site 2 in [1, 10] using the nonlinear objective function. This label is 4, which is selected as the anchor for site 2. Similarly, the anchor for site 1 is 6. At this stage $\mathcal{E}^+ = c(1, 6) + c(2, 4) + 0.5 * |6 - 4| = 3.7$.

- Now, the trust region for $\mathcal{LP}_1$ is shrunk to $[2, 9]$ for both of f_1 and f_2 by reducing the previous trust region diameter by a factor of 2. The solution of $\mathcal{LP}_1$ is $f_1 = 6$ and $f_2 = 6$. The anchor site is 6 for site 1 and 5 for site 2, with $\mathcal{E}^+ = 3.2$.

- Based on $\mathcal{LP}_1$, $\mathcal{LP}_2$ has a new trust region $[3, 8]$ for both f_1 and f_2 and its solution is $f_1 = 6$ and $f_2 = 6$. The anchors' positions do not change at this stage.

- $\mathcal{LP}_3$ has a new trust region $[4, 7]$ for both f_1 and f_2 and its solution is $f_1 = 6$ and $f_2 = 5$. Since LP achieves the upper bound $\mathcal{E}^+$, there is no need to further shrink the trust region and the iteration terminates. It is not difficult to verify that the configuration $f_1 = 6$, $f_2 = 5$ achieves the global minimum. Fig. 2.9 illustrates the proposed successive convexification process method for this example.

Interestingly, for the above example ICM or even GC only finds a local minimum if initial values are not correctly set. For ICM, if f_2 is set to 8 and the updating is from f_1, the iteration will fall into a local minimum corresponding to $f_1 = 9$ and $f_2 = 8$. The Graph Cut scheme based on α-expansion will have the same problem if the initial values of f_1 and f_2 are set to 9 and 8 respectively.

Example 2.4 (A 2-D problem): In this example, the graph template has 4 nodes and we try to locate the object in clutter. Figs. 2.10, 2.11, 2.12 and 2.13 show several selected stages of the convex relaxation iteration and illustrate the trust region shrinkage and refined matching at each stage. Sub-figures in the first rows show the templates and matching result in the target image. The blue dots in the target images are true target points. The second

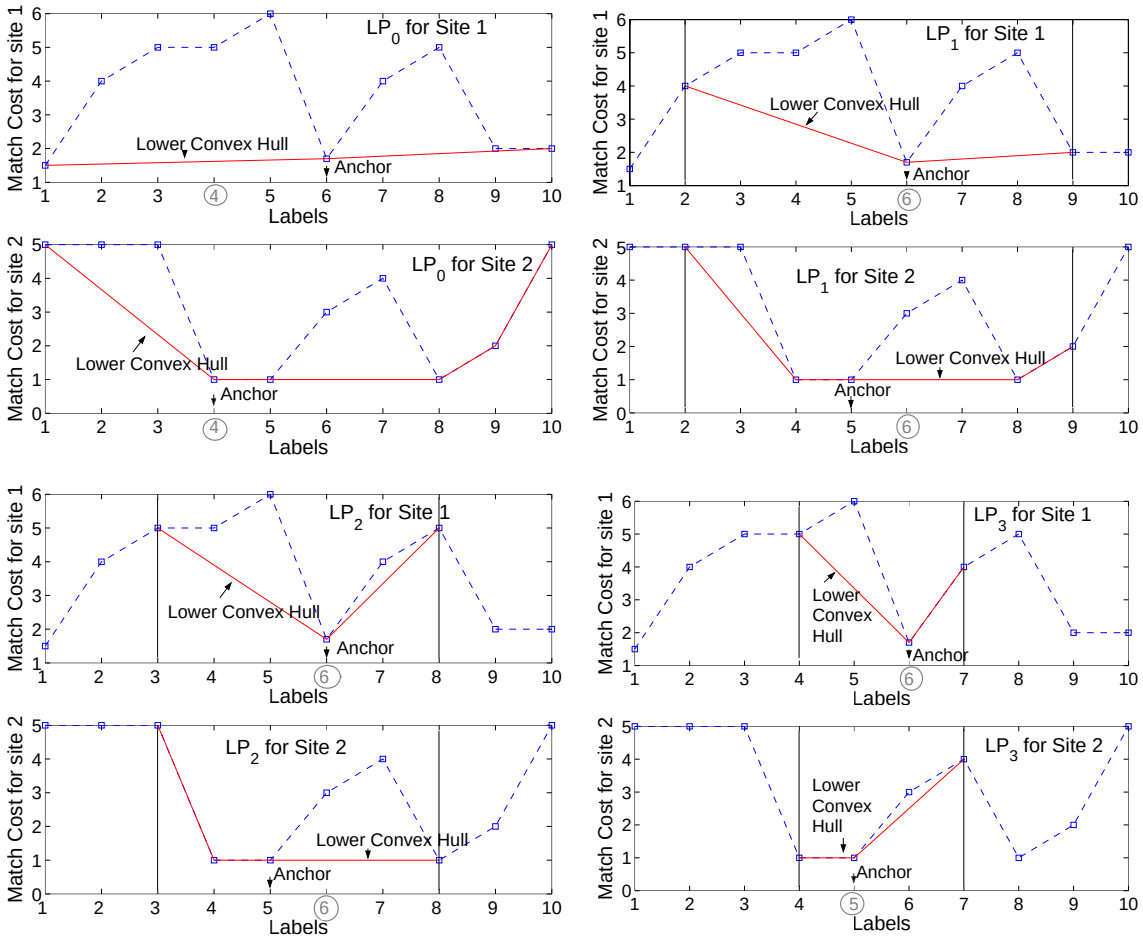


Figure 2.9: Successive convexification LP in 1-D. Labels in circles are LP continuous solutions.

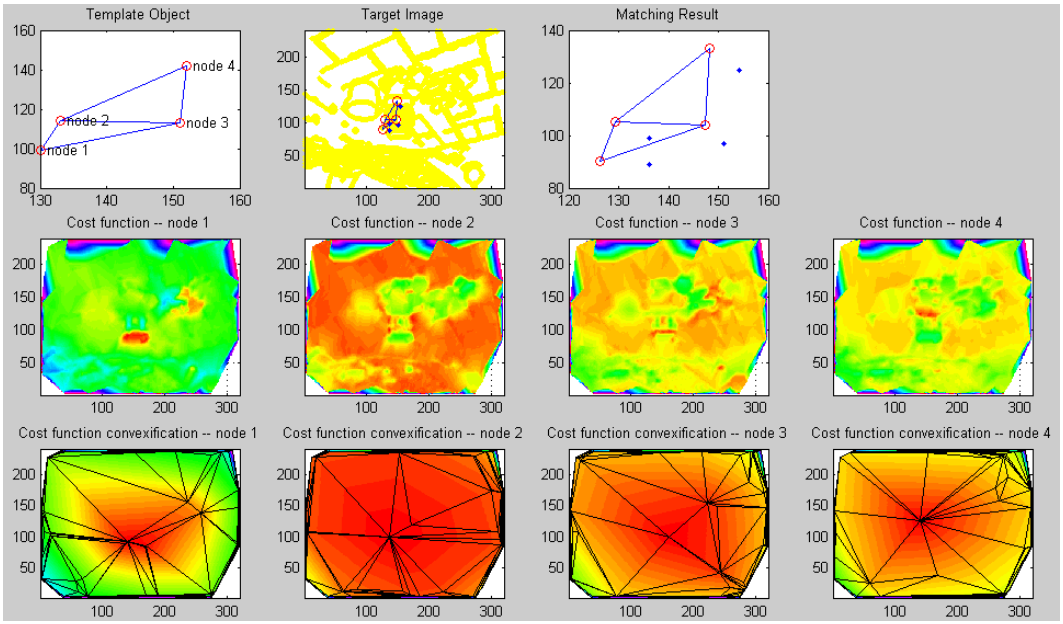


Figure 2.10: Successive convexification LP in 2-D at stage 0. Warmer colors indicate lower values.

rows show labeling cost surfaces for each site. These cost surfaces need only be computed once and remain unchanged during the successive convexification. The last rows show the convexified surfaces in smaller and smaller trust regions. To make the region shrinking process clearer, in calculating the convexified cost surface in a smaller region, we did not remove the labels outside of the trust region; instead, we set their cost at a large number. Calculating lower convex hull is discussed in more detail in Section 2.7. The trust regions are shrunk from the whole target image to 5×5 rectangles in 9 stages. As shown in this example, successive relaxation indeed helps to overcome the coarse approximation problem of single LP. In practical applications, the trust regions can be shrunk very quickly. Typical problems involve 3–5 iterations.

2.5 Complexity of the Successive Convexification Linear Programming

We have proposed a successive convexification method for solving consistent labeling problems with convex regularization terms. As shown in the properties, the convex relaxation

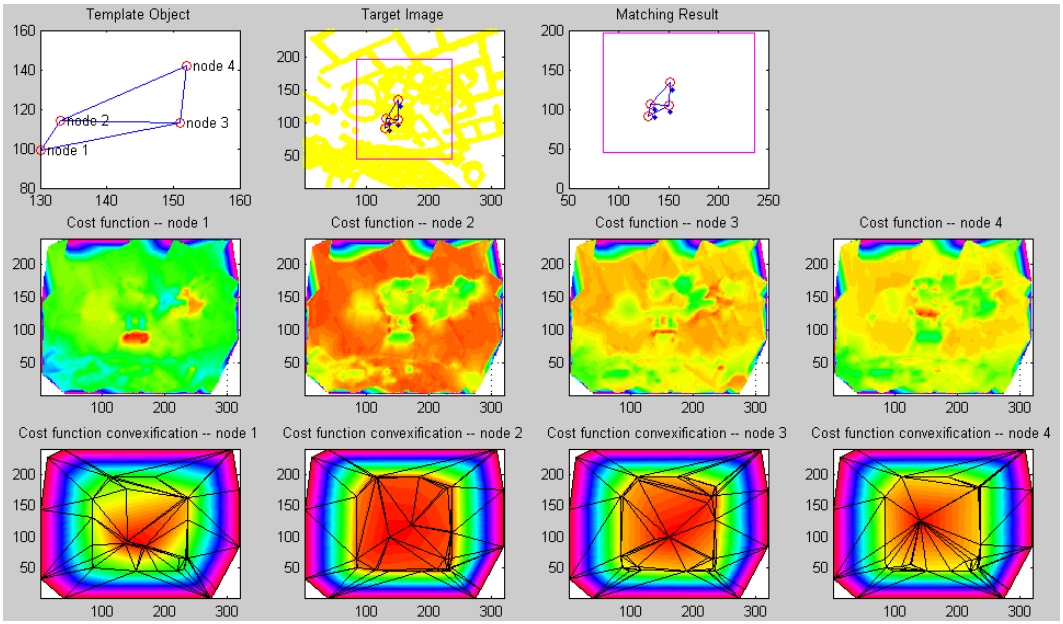


Figure 2.11: Successive convexification LP in 2-D at stage 1. Warmer colors indicate lower values.

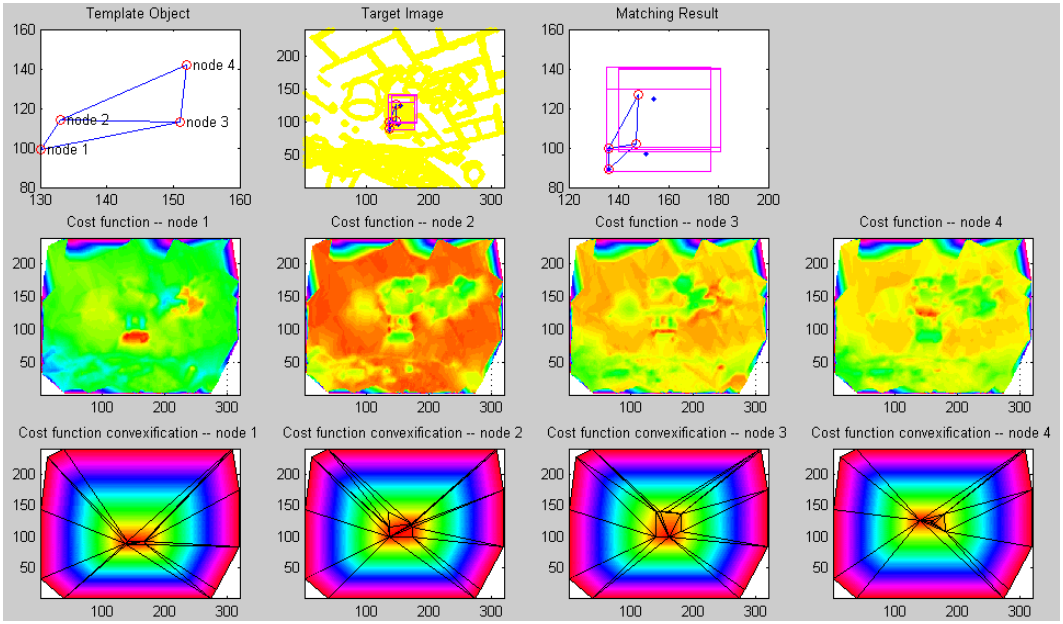


Figure 2.12: Successive convexification LP in 2-D at stage 4. Warmer colors indicate lower values.

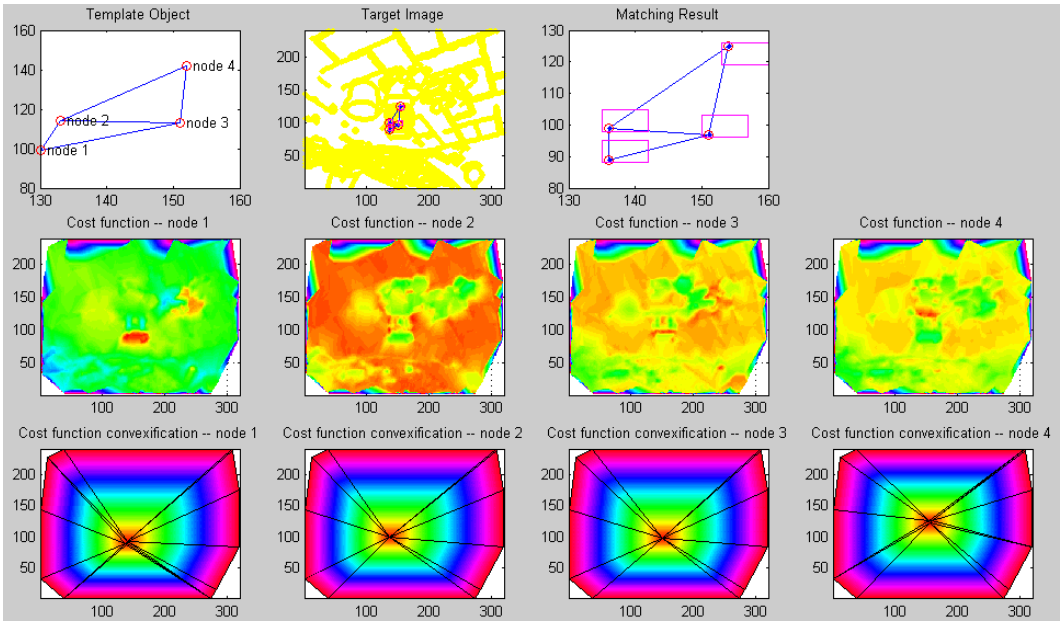


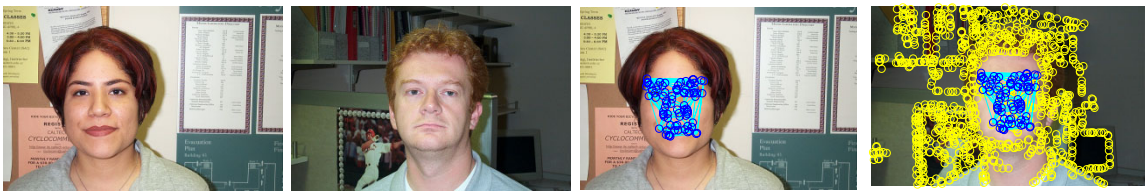
Figure 2.13: Successive convexification LP in 2-D at stage 8. Warmer colors indicate lower values.

only involves basis labels and their labeling costs in the optimization process. The basis labels correspond to vertices of the lower convex hull for labeling cost surface at each site. In labeling applications, for each site, there is only one corresponding label. In most cases, the "best" label has small labeling cost and other labels act as clutter which usually have bigger costs. Adding more clutter labels having higher costs does not change the shape of the lower convex hull and therefore does not change the size of the relaxed convex problems. The number of iterations in the successive convexification is also usually fixed and thus the size of the successive convex programming does not change with the increasing clutter in the ideal case. The complexity to calculate lower convex hull does increase with the number of clutter points but it is light weight in practice. Traditional labeling schemes do not have such a property and they have to involve all the labels in the optimization. In real situations, clutter labels may have equal cost as or even lower costs than that of the true label and this makes the labeling cost surface complex. But in practice, the size of the convex problem increases very slowly with the increase of labels and the convex programming is largely decoupled from the size of label set.

The simplex method has been found to be efficient in applications even though its worst

case complexity is exponential. The complexity of linear programming increases slowly with the number of variables but faster with the number of constraints. The heuristic average complexity [41] is found to be approximately $O(k\log(l))$, where k is the number of constraints and l is the number of variables. In our implementation, the number of constraints $k = 3|S|+2|\mathcal{N}|$ and number of variables $l = \sum_{\mathbf{i}\in S}(|\mathcal{B}_\mathbf{i}|+2)+2|\mathcal{N}| \leq |S|(|L|+2)+2|\mathcal{N}|$, where L is the label set. We assume that the relation topology of source sites is a planar graph and thus $|\mathcal{N}|$ is bounded by $3|S|$, and therefore $k \leq 9|S|$ and $l \leq |S|(|L|+8) \approx |S|\cdot|L|$. The number of LPs involved in successive convexification is usually a constant. We can ignore the reconvexification complexity. Therefore, an estimate of the average complexity of successive reconvexification linear programming is $O(|S|\cdot(\log|L|+\log|S|))$. Experiments also confirm that the average complexity of the proposed optimization scheme increases more slowly with the size of label set than previous methods such as Graph Cut whose average complexity is linear with respect to $|L|$ and BP whose average complexity is proportional to $|L|^2$. Moreover, because successive convexification only uses basis labels, the basis label set is much smaller than L.

In a standard image matching application as shown in Fig. 2.14, we would like to align the template face with the target face in a cluttered background. We use log-polar features on the distance transform of edge maps. We will discuss more about image features in the section on image matching. The sites are feature points on the template face and the labels are randomly selected edge points in the target image. We wish to test the complexity increase of the linear programming with increasing of the number of labels and sites. Fig. 2.15 (a) shows test cases in which we increase the number of target points (labels) while keeping the number of sites fixed to be a constant. As expected, the linear program's size which is characterized by its variable's number and constraint's number is almost decoupled from the number of labels. The simplex iterations also almost keep constant even if we greatly increase the number of labels. Surprisingly, the linear program in fact could become smaller when the number of labels increases (as shown in Fig. 2.15 (a)). This is possible because as the density of target features increases some good matches may be found, which therefore may reduce the lower convex hull's vertices number. Fig. 2.15 (b) shows the scenario when we increase the number of template features points (sites) and keep the number of labels to the constant of 1000. In this case, the number of LP variables increases faster than that of the constraints. The simplex iterations increase with nearly the same trend as the constraint number but much slower than that of the variable number.



(a) Template Image (b) Target Image (c) Template Mesh (d) LP Matching Result

Figure 2.14: Matching faces in clutter. Light circles in (d) show all the feature points in target image. Images (a) and (b) ©Caltech, 2006, by permission.

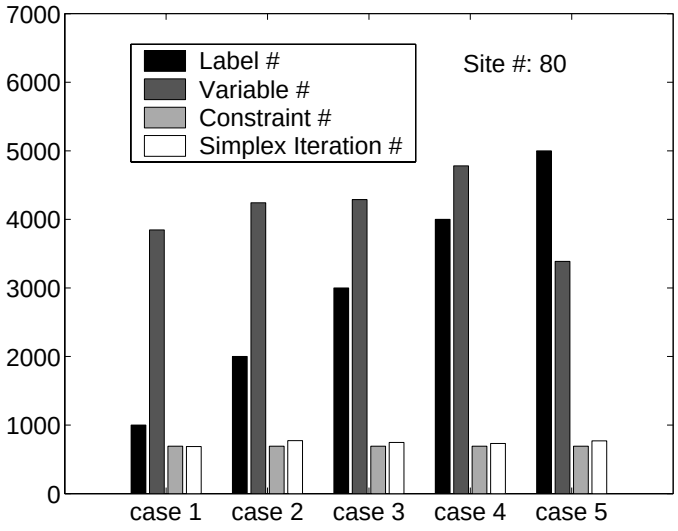
Table 2.1: Running Time Comparison (2.6GHz PC).

	ICM (50 passes)	SC-LP (1Pass, typical 3-5)	BP (1Pass, typical 4-8)
Running time	1.67s	1.0s	99.2s
Success	No	Yes (in 3 passes)	Yes (in 4 passes)

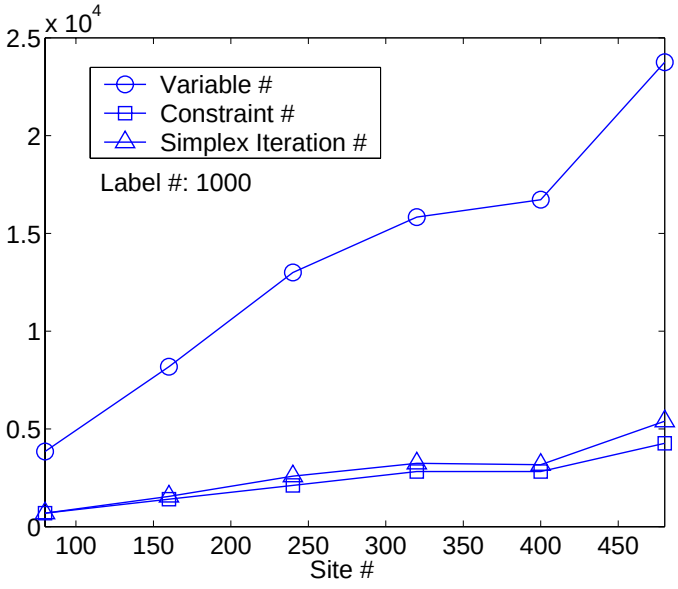
This complies with the model we used in the complexity analysis.

We further compare the running time for the case of 80 sites and 1000 labels, with different methods. The result is shown in Table 2.1. The greedy method ICM iterates 50 times. It takes only 1.67s, but does not locate the object correctly. BP does find the right matching in 4 iterations. But it is much slower than the successive convexification scheme. This example is a typical case for the running times of different methods.

BP can be sped up by embedding displacement vectors into grids and computing function lower envelop with a method similar to distance transform [6]. This scheme has linear complexity over the number of labels (displacement vectors). But it has to trade time complexity with high spatial complexity. Even we only have sparse candidate points in the target image, we still need to store values at grids corresponding to all the possible motion vectors when computing the function lower convex hull. We therefore need to store a $(2W) \times (2H)$ float map for each neighboring site pair, where W and H are target image's width and height respectively. Even using a small number of template points, in image matching its memory usage may go up to giga-bytes. The high memory usage limits the deployment of this scheme. Multi-resolution schemes can also be used to reduce the complexity of BP for large label set problems. We can group multiple target points into coarser labels and subdivide the groups iteratively for matching refinement. In one implementation, we subdivide the target images into blocks. For each site, the labeling cost corresponding to a block is the average cost (or the minimum cost) of all the target feature points in the



(a)



(b)

Figure 2.15: Complex of SC-LP.

block. The candidate target blocks only contain the ones which has one or more target feature points. BP can be applied to the labeling problem in this coarse level first. Based on the labeling result in the coarse level, the target points are restricted into smaller areas in refinement matching. We thus can split the blocks recursively until we go to the pixel level. Multi-scale methods may degrade the matching result. As blocks become bigger, the distinguishability between blocks becomes weaker. When the target image's resolution is reduced by more than 8 times, the result degrades rapidly. Because of this constraint, multi-scale BP is still much slower than the proposed method without multiscale speedup for large label set labeling problems.

2.6 Successive Relaxation for General Convex Regularization Labeling

In the previous sections, we present a successive relaxation method for solving labeling problems with L_1 regularization terms. The basic idea is to convexify the labeling cost surfaces for each site and formulate a convex programming problem by using only the basic labels. To improve the approximation accuracy, we gradually shrink the trust region for each site. This process iterates until the trust regions become small. In fact, this successive convexification process is not just restricted to problems with L_1 regularization terms. Theoretically, all convex norm regularization consistent labeling problems can be relaxed and solved with the proposed scheme.

As an example, we show how to convert a consistent labeling problem with L_2 norm regularization terms into a convex quadratic programming (QP) problem. We still use the nonlinear formulation Eq. 2.1 but use the L_2 norm in the regularization term. Similar to the L_1 norm case, for each $\mathbf{s}$ on the template, we replace the matching cost $c(\mathbf{s}, \mathbf{f_s})$ with a linear combination of the *basis labeling* costs: $c(\mathbf{s}, \mathbf{f_s}) \approx \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} \cdot c(\mathbf{s}, \mathbf{t})$, where $\mathcal{B}_\mathbf{s}$ is the set of basis labels for $\mathbf{s}$ and $\xi_{\mathbf{s},\mathbf{t}}$ are non-negative coefficients. These points serve as a basis such that $\mathbf{f_s} = \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} \cdot \mathbf{t}$, with the constraint $\sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} = 1$ for each $\mathbf{s}$. The convex quadratic relaxation problem is:

$$\min \left\{ \sum_{\mathbf{s}\in S} \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} c(\mathbf{s}, \mathbf{t}) + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} [(u_\mathbf{p} - u_\mathbf{q})^2 + (v_\mathbf{p} - v_\mathbf{q})^2] \right\}$$

$$
\begin{aligned}
s.t. \quad \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} &= 1, \forall \mathbf{s} \in S \\
u_\mathbf{s} &= [\sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} \cdot x(\mathbf{t})] - x(\mathbf{s}), \\
v_\mathbf{s} &= [\sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{t}} \cdot y(\mathbf{t})] - y(\mathbf{s}), \forall \mathbf{s} \in S \\
\xi_{\mathbf{s},\mathbf{t}} &\geq 0, \forall \mathbf{s} \in S, \forall \mathbf{t} \in \mathcal{B}_\mathbf{s}
\end{aligned}
$$

in which we denote functions $x(\mathbf{s})$ and $y(\mathbf{s})$ as extracting the x and y component of site $\mathbf{s}$. If we constrain the variables $\xi_{\mathbf{s},\mathbf{t}}$ to be binary (0 or 1) and $\mathcal{B}_\mathbf{s}$ includes all the candidate labels for site $\mathbf{s}$, the optimization problem is exactly equivalent to the original non-convex labeling problem. The convex QP relaxation shares similar properties with the LP relaxation for L_1 norm problems. In fact, these properties also apply to other labeling problems with convex norm regularization terms.

Property 2.7: When $\mathcal{B}_\mathbf{s}$ contains all the candidate labels for $\mathbf{s}$, and the continuous extension cost function $c(\mathbf{s}, \mathbf{t})$ is convex with respect to $\mathbf{t}$, $\forall \mathbf{s} \in S$, the QP exactly solves the continuous extension of the discrete labeling problem.

Property 2.8: If the cost function $c(\mathbf{s}, \mathbf{t})$ is highly non-convex with respect to $\mathbf{t}$ for each site $\mathbf{s}$, the quadratic programming formulation solves the continuous extension of the reformulated discrete labeling problem, with $c(\mathbf{s}, \mathbf{t})$ replaced by its lower convex hull for each site $\mathbf{s}$.

Property 2.9: We also need only include the basis set $\mathcal{B}_\mathbf{s}$ comprised of the vertex coordinates of the lower convex hull of $c(\mathbf{s}, \mathbf{t})$, $\forall \mathbf{s} \in S$, into the optimization process.

This makes the complexity of the scheme increase very slowly with the the number of labels. To improve the relaxation, we can use similar trust region shrinkage method as the successive LP relaxation. In each iteration, we need to reconvexify the cost surface and construct a new quadratic programming problem. Convex quadratic programming can be efficient globally optimized with Prime-Dual interior method.

2.7 About the Lower Convex Hull

One of the basic operations in successive convexification is to compute the lower convex hull of a labeling cost surface. Computing a convex hull has been intensively studied. There are many efficient algorithms such as gift wrapping and incremental schemes [84]. Computing a lower convex hull can be done in a similar way. In fact, we can directly compute the convex

hull of the labeling costs and then remove the faces of the hull whose norm has a sharp angle with the z (or label cost) axis. The problem with such an approach is that it wastes a lot of time in calculating the upper hull faces. To solve this problem, we compute the lower convex hull using the following incremental scheme. This method is a modification of the incremental scheme for a convex hull.

The initial lower convex hull is set as follows. For labeling applications, the labels are in a finite region. For instance, in image matching, the labels are the target image points and locate inside the range of the image. We can build an initial lower convex hull by introducing 4 extra labels located at $(x_{\min} - \delta, y_{\min} - \delta), (x_{\min} - \delta, y_{\max} + \delta), (x_{\max} + \delta, y_{\max} + \delta)$ and $(x_{\max} + \delta, y_{\min} - \delta)$, where δ is a small positive number. We then set the labeling costs for the 4 extra labels at a large value, which is much greater than the maximum labeling cost. We then randomly select one label and build the initial lower convex hull. An initial lower convex hull is illustrated in Fig 2.16 (b). After the initialization, we incrementally add more points and update the hull sequentially.

If a newly added point is above the lower convex hull, it is labeled not on the hull. Testing whether a point is above the lower convex hull can be done by computing the signed tetrahedron volume spanned by the point with each triangle patch. The volume can be computed with

$$\det\left(\begin{bmatrix} v_{1x} - p_x & v_{1y} - p_y & v_{1z} - p_z \\ v_{2x} - p_x & v_{2y} - p_y & v_{2z} - p_z \\ v_{3x} - p_x & v_{3y} - p_y & v_{3z} - p_z \end{bmatrix}\right)$$

where (v_{ix}, v_{iy}, v_{iz}) $i = 1..3$ are vertices of the triangle patch and (p_x, p_y, p_z) is the point to be tested. (v_{ix}, v_{iy}, v_{iz}) $i = 1..3$ are counter-clockwise, looking from above the lower convex hull. The volume is negative if testing point is above the patch plane. Thus, if all the volumes spanned by the point and triangle faces are negative, the point is above the current lower convex hull and labeled as not on the hull. The point is denoted to be able to see a triangle patch if it is below a patch. For a newly added point that is under the current lower convex hull, we need to update the lower convex hull by removing the faces this point can see and make new triangle faces from the point to the visible-invisible boundary edges. If a visible face has an edge at the boundary of the lower convex hull, we also need to make a new face spanned by the point with this edge. We then remove all the edges between two adjacent visible faces. Finally, we can remove the 4 extra points and triangle patches with

vertices incident at the 4 corner points. Figs. 2.16 (c), (d) and (e) show the incremental procedure of computing the lower convex hull for 51 randomly generated 3-D points. In the last step, as shown in Fig. 2.16 (f), the triangles attached to the extra four corner points are removed. The vertices of the lower convex hull are then used as basis labels for constructing convex programs.

The above incremental algorithm has a complexity of $O(n^2)$, since for each point we need to process all the edges and faces which are bounded by a linear function of the 3-D point number n. This algorithm can be improved to $O(n \log n)$ with a divide-and-conquer method by partitioning the points into smaller groups and combining with a simple merging procedure. For 6,500 labels, typical running time of lower convex hull for each site in a 2.6GHz PC is 0.16s.

2.8 Labeling Problem with Higher Dimensions

In the previous sections, we study the successive convexification scheme in the context of image matching, in which each label is a 2-D point in the target image. For n-D labeling problem with L_1 regularization terms, we have the following general LP relaxation

$$\min \left\{ \sum_{\mathbf{s} \in S} \sum_{\mathbf{j} \in \mathcal{B}_\mathbf{s}} c(\mathbf{s}, \mathbf{j})\, \xi_{\mathbf{s},\mathbf{j}} \; + \sum_{\{\mathbf{p},\mathbf{q}\} \in \mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{n} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}) \right\} \tag{2.3}$$

with constraints:

$$\sum_{\mathbf{j} \in \mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}} = 1 \;, \forall \mathbf{s} \in S$$

$$\sum_{\mathbf{j} \in \mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}}\, \phi_m(\mathbf{j}) = f_{\mathbf{s},m}, \;\; \forall \mathbf{s} \in S \,, \;\; \phi_m(\mathbf{j}) = (m\text{th component of } \mathbf{j}) \;\; , m = 1, n$$

$$f_{\mathbf{p},m} - f_{\mathbf{q},m} - \phi_m(\mathbf{p}) + \phi_m(\mathbf{q}) = f^+_{\mathbf{p},\mathbf{q},m} - f^-_{\mathbf{p},\mathbf{q},m} \,, \forall \; \{\mathbf{p},\mathbf{q}\} \in \mathcal{N} \,, m = 1, n$$

$$\xi_{\mathbf{s},\mathbf{j}}, \; f^+_{\mathbf{p},\mathbf{q},m}, \; f^-_{\mathbf{p},\mathbf{q},m} \geq 0$$

For 2-D problems, successive convexification replaces the original labeling cost surface with its lower convex hull. In fact, when we say lower convex hull we implicitly assume that the lower convex hull is with respect to the axis of labeling cost. For 2-D cases, we combine x and y coordinate of a label with the labeling cost and form a 3-D point. The above framework can be easily extended to a n-D case, in which each label is a n-D point defined

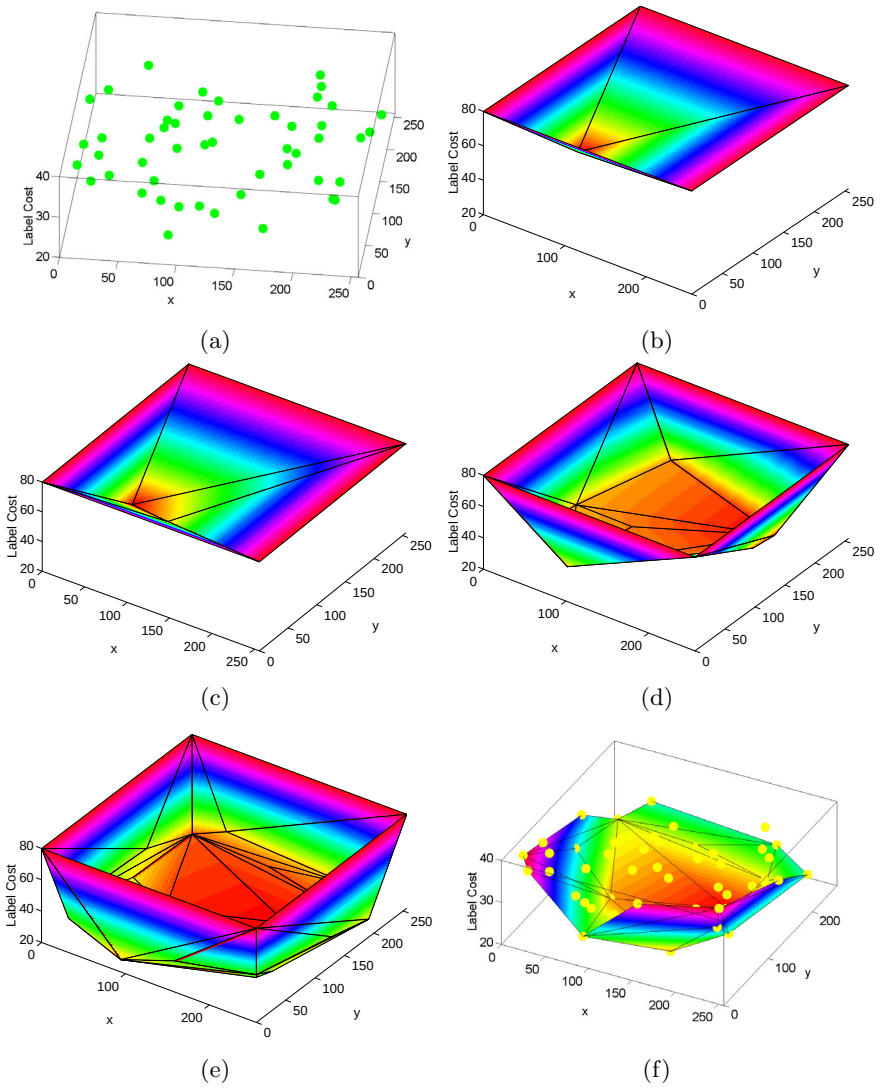


Figure 2.16: Incremental method for lower convex hull. (a): Labeling cost at each position; (b): Initial lower convex hull; (c): Lower convex hull after 2 more points are included; (d): Lower convex hull after adding 10 more points; (e): Lower lower convex hull after adding 50 more points; (f): Final lower convex hull with 3-D points. Warmer colors indicate lower values.

in some metric space. For a convex regularization labeling problem, the convexification scheme still applies. We can follow a procedure very similar to the 2-D case to find the basic labels: we augment each label with another dimensionality of matching cost and form $(n+1)$-D vectors. We then find the lower convex hull of the $(n+1)$-D points with respect to the labeling cost axis. The basis labels correspond to the n-D sub-vectors of the $(n+1)$-D lower convex hull vertices. These n-D sub-vectors are obtained by stripping off the last labeling cost element. The lower convex hull can also be computed efficiently for the higher dimensional cases.

The n-D L_1 norm labeling problems share similar properties with the 2-D cases. For example, Property 2.1 - Property 2.4 can be directly applied to higher dimensional problems. We can follow the induction of Property 2.5 and get:

Property 2.10: *Using simplex method, there will be at most $(n+1)$ nonzero-weight basis labels for each site in n-D labeling problem.*

One of the applications of higher dimensional labeling is volume data registration, such as MRI, CT, etc. in medical imaging. The proposed successive convex labeling scheme can be directly applied to these applications. Higher dimensional problems with other convex regularization terms such as L_2 norm can also be solved using successive convexification schemes similar to the L_1 norm problems.

2.9 Labeling Problem with Nonconvex Smoothness Terms

We have studied successive convexification based on the assumption that smoothness terms are convex. In the following, we show that we can extend successive convexification to approximate labeling using nonconvex smoothness terms. Nonconvex smoothness terms using truncated L_1 or truncated L_2 norms have been found to show better discontinuity preserving properties in matching applications such as stereo.

Similar to GNC, initially, we simply replace the nonconvex distance function, e.g. truncated L_1 distance, with a convex L_1 distance function that has a small slope and is bounded by the image dimension. When shrinking trust regions, we can choose smoothness coefficients adaptively so that the L_1 distance function approximates the nonconvex distance function in smaller trust regions. We estimate the range of the matching vector difference for each neighboring site pairs when shrinking the trust regions. The slope of the L_1 distance, controlled by the coefficient $\lambda_{\mathbf{p},\mathbf{q}}$, is then adjusted to fit the nonconvex distance in

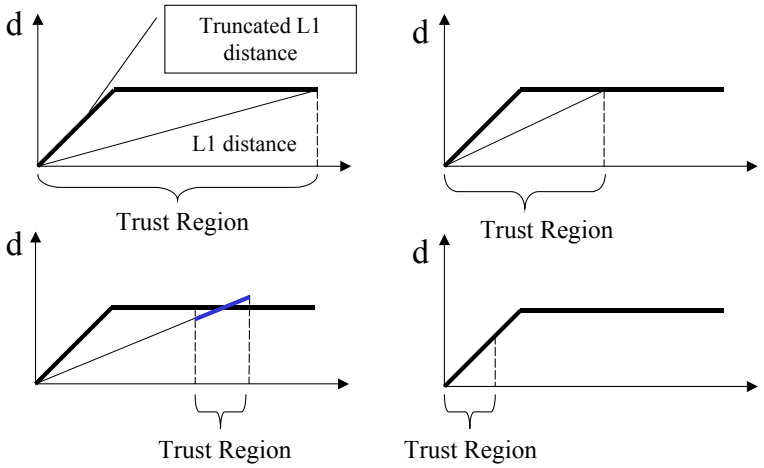


Figure 2.17: Approximating nonconvex distance functions in Successive Convexification.

the estimated range. Fig. 2.17 illustrates approximating non-convex distance function, the truncated L_1 distance function, in a specific region using a L_1 distance function with a suitable slope. The top-left subfigure in Fig. 2.17 shows a early stage of successive convexification, in which approximating the truncated L_1 distance with a L_1 can have large error. When the trust region becomes smaller, as shown in the top-right subfigure of Fig. 2.17, the distance approximation improves. When the trust region becomes small, the approximation becomes more accurate as shown in the bottom subfigures of Fig. 2.17.

As the trust regions become smaller, the nonconvex distance function can be more accurately approximated by the L_1 distance function. Such a scheme helps preserving large discontinuities in matching. Comparison with BP using truncated L_1 smoothness term is shown in the experiment section.

2.10 Deformable Object Matching

In this section, we consider the application of the proposed general matching scheme for image matching problems. In deformable object matching, we would like to find the pointwise correspondence between a template and a target object. Usually, the target object will be an image with a lot of background clutter. In this section, we illustrate the application of the proposed successive convexification method in both color object and binary object matching. Deformable template matching forms the middle level processing of most object

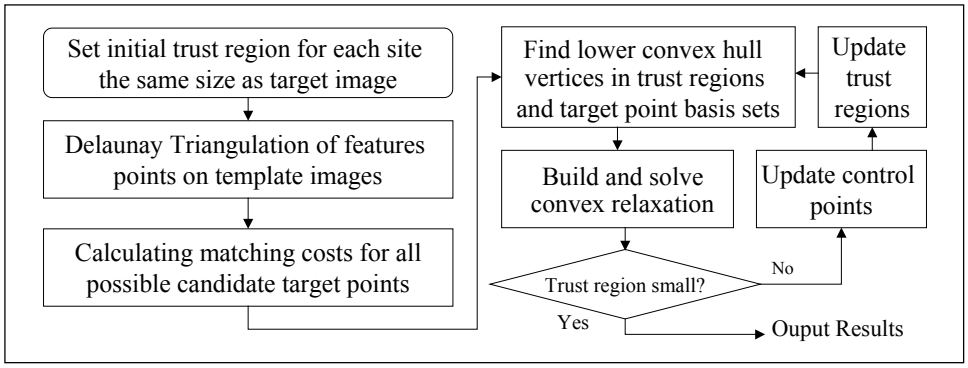


Figure 2.18: Object Matching using Successive Convexification.

recognition applications.

Fig. 2.18 is the diagram of the proposed object matching scheme. The proposed scheme does not limit the type of features selected. Here, we use simple local features to illustrate the usefulness of the method. For color images, we take as features in the template image the square pixel blocks centered on source pixels. For binary images, we first convert to grayscale via a *distance transform*, where the intensity of a pixel is proportional to its distance to the nearest foreground binary pixel. We define features in a binary image as the small blocks in such distance transform images. This feature is found to be superior to the traditional local tangent orientation or curvature, in that it is more stable and there are efficient algorithms for the distance transform itself.

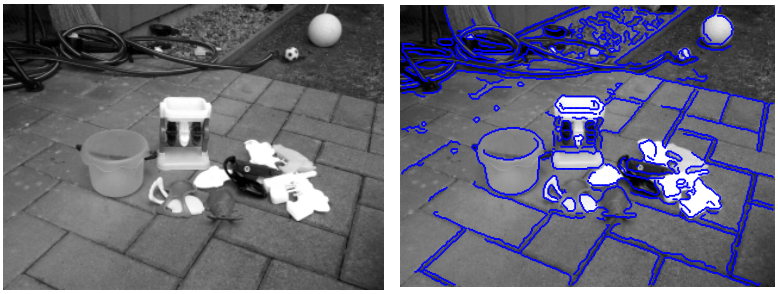
Feature points are randomly selected in the source image. To reduce complexity, the feature points can be selected centered on the edges of the source and target images. The neighbor relation $\mathcal{N}$ is defined by the edges of the graph resulting from Delaunay triangulation of point set S on the template. For the target image, depending on the application, all the pixels in the searching window or all the edge pixels in the target image are used as matching candidates. Fig. 2.19 illustrates an object matching example. In this example, the template and target images are grayscale. A region of interest containing the toy object is manually selected in the template image. Then 5% of edge pixels in the region of interest are extracted as the template features points. Delaunay triangulation is applied to these feature points and the graph structure is generated as shown in Fig. 2.19 (b). All the edge pixels in the target image as shown in Fig. 2.19 (d) are used as matching candidates. The successive convex matching with trust region of the whole target image is shown as Fig. 2.19 (e). The final matching with 4 iterations is shown as Fig. 2.19 (f). As a comparison, the greedy

scheme matching result is also shown in Fig. 2.19 (g). The greedy scheme fails to locate the target in the image because of the similarity of features in the background.

We often need to test the degree of matching. We use such a measure to select the best template in multi-template problems. We use the following quantities to measure the difference between the template and the matching object. First we define measure P to be the average of pairwise length changes from the template to the target. To compensate for any global deformation, a global affine transform $\mathcal{A}$ is first estimated based on the matching and then applied to the template points before calculating P. The length changes are further normalized by the template edge lengths. The second measure is the average warped template matching cost M, which is taken to be the difference of the target image or distance transformation and the warped reference image or distance transformation in the region of interest. The warping is based on a cubic spline. The total matching cost is simply defined as $M + \alpha P$, where α typically has a value from 0.1 to 0.5. Experiments show that only about 100 randomly selected feature points are needed in calculating P and M.

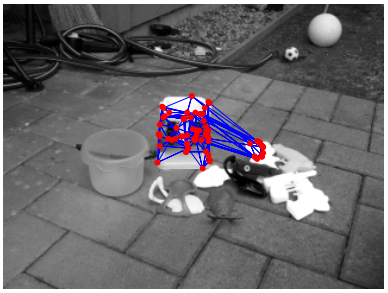
For some problems, the template undergoes scale and rotation changes. To simplify the problem, we decompose the geometrical transformation of the template into two cascaded transformations: a global transformation $\mathcal{G}$ and a local deformation $\mathcal{D}$. The global transformation is shared by all the template sites while the local deformation may be different for each site. In this application, the matching cost $c(.)$ is a function of the source pixel (site), the target pixel (label), and the global transformation (such as scaling and rotation). Once $\mathcal{G}$ is fixed, the problem is reduced to the consistent labeling problem discussed in the previous sections and we can apply the proposed successive convexification scheme to solve $\mathcal{D}$. We limit $\mathcal{G}$ to rotation and scaling, but more complex geometrical transformations such as affine transformations could be applied. However, affine transformations will increase complexity and not have a big advantage over similar transformations for general object matching applications [25]. We found that very sparse quantization is enough for estimating the global transformation $\mathcal{G}$. In our scheme, the quantization interval for rotation is 45 degrees and scale is quantized into levels of 0.5, 0.75, 1, 0.5 and 2. We can use the average matching cost of successive convex matching in the largest trust region as the score to determine $\mathcal{G}$.

(a) Template image (b) Template mesh



(c) Target image (d) Target feature points

(e) SC-LP Initial match (f) SC-LP matching result



(g) Greedy matching result

Figure 2.19: An example of object matching.

2.11 Experimental Results

2.11.1 Testing on Synthetic Images

In this experiment we compare the performance of successive convexification linear programming (SC-LP) with BP and ICM for binary object detection in clutter. GC has been most applied to motion and stereo and therefore not included in the comparison. In our experiments, the templates are generated by randomly placing 100 black dots into a 128×128 white background image. A 256×256 target image is then synthesized by randomly translating and perturbing the block dot positions from those in the template. The random perturbation ranges are set to be 5 and 10 pixels respectively in two experiments. 50, 100 or 150 random noise dots are then added to the target image to simulate background clutter. There are thus 6 situations to be tested and in each experiment we generate 100 template and target images. In this experiment, we match the graylevel distance transformation of the template and target images. Fig. 2.20 compares results using the proposed SC-LP matching scheme with using BP and ICM. The matching error histograms show error distribution of different methods. In this experiment, all the methods use the same energy function. The SC-LP has similar performance as BP and much better than the greedy scheme of ICM in cases of large distortion and cluttered environments. SC-LP has much less complexity than BP when the number of labels exceeds one hundred.

In another experiment, we test random object matching by using L_2 norm. The experiment setting is the same as the previous one for SC-LP matching. The feature used in this experiment has a bigger context by using the log-polar transform of distance transform images. In each outlier and perturbation setting, we randomly generated 100 pairs of testing images and compared the results. Fig. 2.21 shows the matching error distribution for different methods, in several outlier and perturbation settings. In low distortion and low outlier cases, all three methods have similar performance. But when the outliers and distortion increase, greedy schemes degrade rapidly. The proposed scheme works as well as BP in this experiment and much more efficient.

In a different experiment setting for matching random dots, we use a truncated L_1 distance function in the regularization term. In addition to random disturbance of 5 pixels, there is a 30-pixel displacement in x direction and a shift of 5 pixels in y direction for the right half of the template points. The nonconvex smoothness term helps to preserve the discontinuity in matching. Using SC-LP, we adaptively change the smoothness coefficients

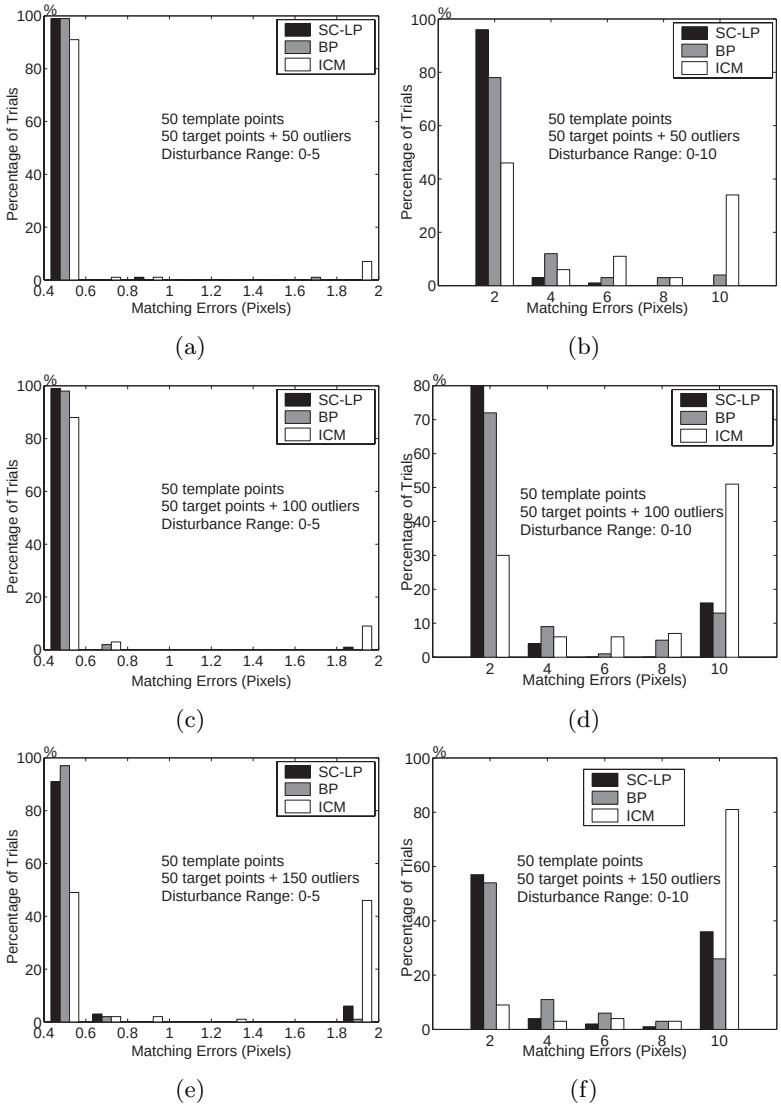


Figure 2.20: Histogram of errors using SC-LP, BP and ICM.

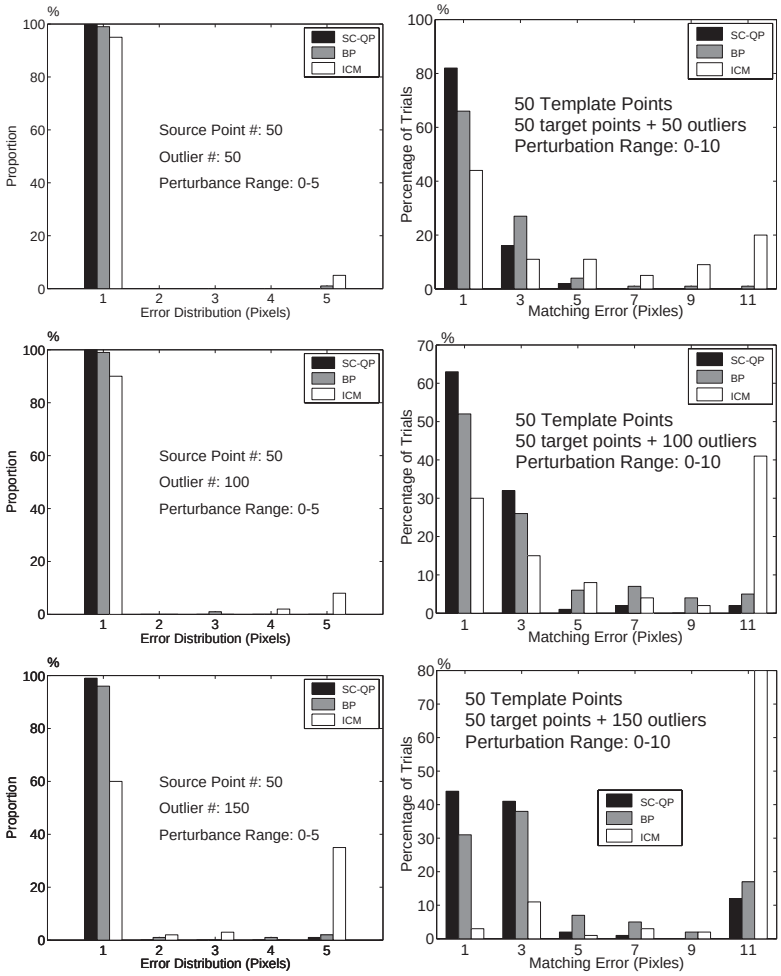


Figure 2.21: Histogram of errors using SC-QP, BP and ICM.

Table 2.2: Comparative results for random dot matching with large discontinuities

	SC-LP	BP	ICM
Mean Absolute Error	6.3227	7.4985	56.8322
Standard Deviation	5.5764	7.4985	12.6827

so as to approximate a truncated L_1 distance function which has a linear part with unity slope and a turning point at 15 pixels. The random dot template is composed of 100 points and 50 noise points are added to the target image. We repeat the experiment for 100 times. Different methods in this experiment use the same energy function. The mean matching errors and standard deviations of the testing methods are shown in Table 2.2. The proposed scheme and BP yield much better results than greedy methods.

2.11.2 Testing on Real Images

Color object matching examples using SC-LP are shown in Fig. 2.22. Fig. 2.22 (a) shows the template mesh for a toy, with feature points randomly selected on the edge map of the object. In this experiment, the global transform $\mathcal{G}$ is first estimated in the discretized global transform space with SC-LP in the largest trust region. Rotation is sampled between 0° and 360° at 45° intervals. The scale is sampled in $[0.5, 2]$ with quantization levels 0.5, 0.75, 1, 1.5 and 2. The interpolated average matching cost surface for $\mathcal{G}$ is shown in Fig. 2.22 (b), with optimal rotation angle θ and scale κ of 45° and 1.5 respectively. Local deformation is further estimated by successive convexification. The final matching result is shown in Fig. 2.22 (c). In Fig. 2.22 (d, e), a toy dog is matched in a very cluttered background. This is a difficult target in that there are many features in the background similar to those on the toy. The proposed method finds the correct global transform and the correct matching. Further results for color object matching are shown in Figs. 2.22 (g, i, k). Figs. 2.22 (j, k) show matching where large occlusion is involved. Fig. 2.23 shows an example for binary object matching using the distance transform image patches as features. In this example, SC-LP performs robustly in strong background clutter.

An object localization result using successive convexification quadratic programming (SC-QP) is shown in Fig. 2.24. Log-polar transform feature for color image is used in this experiment, which allows sparser sampling in the target image. The leaf in the experiment is an object with very little texture. The boundary edges of the leaf are the features that are used to locate the object. Because of the background changes from the template

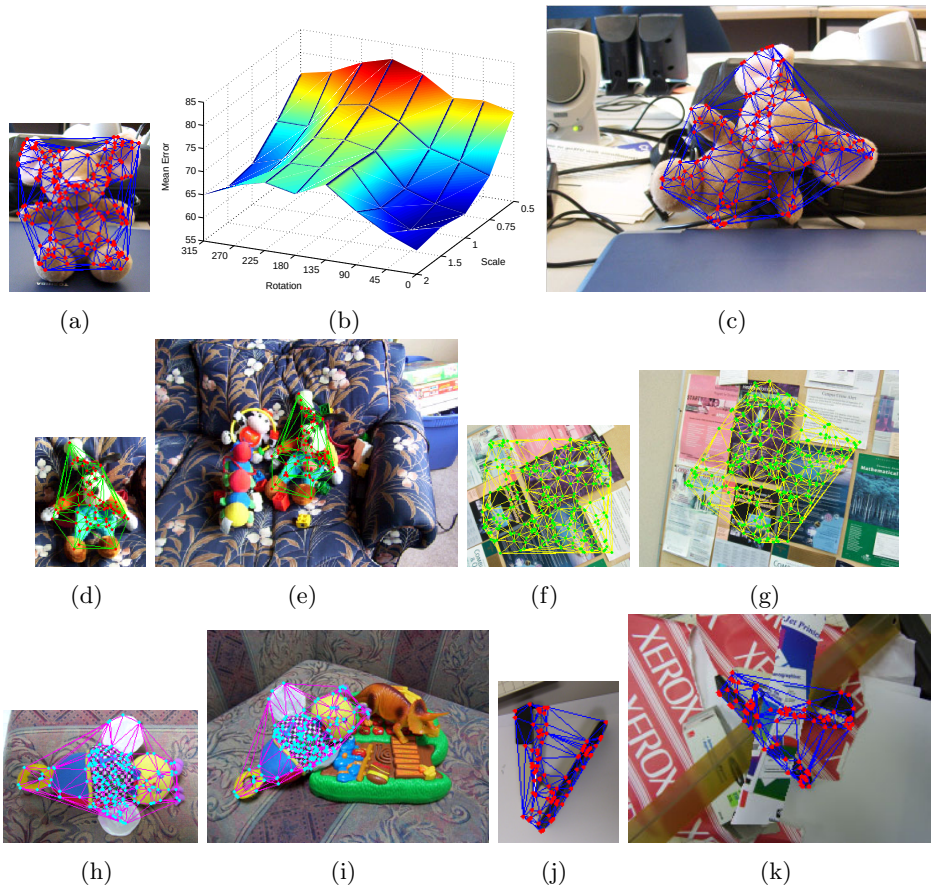


Figure 2.22: Color object matching. (a): Template mesh; (b): Interpolated average matching cost surface for $\mathcal{G}$: $\theta^* = 45°, \kappa^* = 1.5$; (c): Matching result with the SC-LP method. (d, f, h, j): Four other template meshes; (e, g, i, k): Matching results.

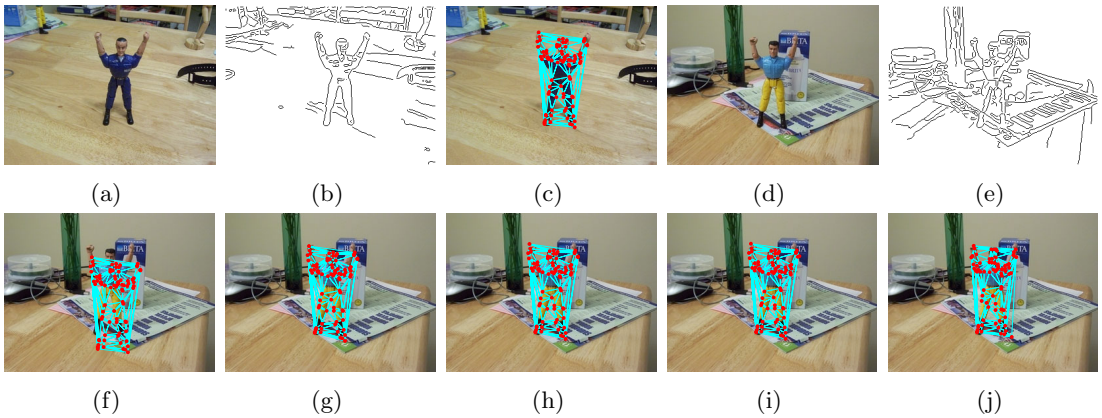


Figure 2.23: Binary image matching. (a, b, c): Template image, edge map and template mesh; (d, e): Target image and edge map; (f-j): Different stages of SC-LP matching.

image to target image, these boundary features are also distorted considerably, which makes invariant feature based schemes fail to find the correct corresponding points. We use about 100 randomly selected edge points on the template and 1000 randomly selected edge points on the target image, as shown in Figs. 2.24 (a) and (b). The matching scores for discretized scales and rotations based on the largest trust region SC-QP are shown in Fig. 2.25 (c). The minimum cost rotation and scale estimations are correctly 45 degrees and 1.25 respectively. SC-QP is then applied to locate the object in the target image. Fig. 2.24 (d) is the initial matching and Fig. 2.24 (e) shows the final matching result. Other object localization results are shown in Fig. 2.25, Fig. 2.26 and Fig. 2.27. In the experiments locating the toy and the hand in images, we use a smaller feature context and denser point candidates (about 8000 target points) in each target image, to increase the reliability of matching in complex backgrounds. BP becomes quite slow for such a large number of target points due to its $O(n^2)$ complexity with respect to the number of target points. ICM is not able to locate the targets correctly. For such large label-set matching problems, the proposed successive convexification QP scheme can efficiently locate the target objects in seconds, using an automatically generated template mesh.

As a different test of the usefulness of the proposed scheme for object detection, we carried out a face detection experiment, using the Caltech database. We use SC-LP in the experiment. Each tested image contains one face. After removing several small repeated images and drawings, these comprise a total of 431 test images. One randomly chosen image is used as a template, testing on the others. We chose to match only the chromaticity of the images, which is more illumination invariant but also has more ambiguities. An ICM-based scheme has a correct matching ratio of 41%, while the proposed scheme yields 98.1% correct. The matching faces, showing the bounding box of the match, are displayed in Fig. 2.28. We also experimented on matching car-back images from the Caltech database. The experiment is also based on SC-LP. We use a randomly chosen image as the template and try to locate the object in other images, based on the grayscale distance transform of the Canny edge maps. Of 119 cars, the matching success rate is 82.4%. We show some matching results in Figs. 2.29 (g) and 2.29 (h). Simple ICM schemes work poorly, with only few correct matches. Another object detection result is shown in Fig. 2.30, matching leaves with SC-LP. In this experiment, we again use the distance transform of the edge maps for matching. We use 6 templates, with success for 181 targets out of 185 matching tests.

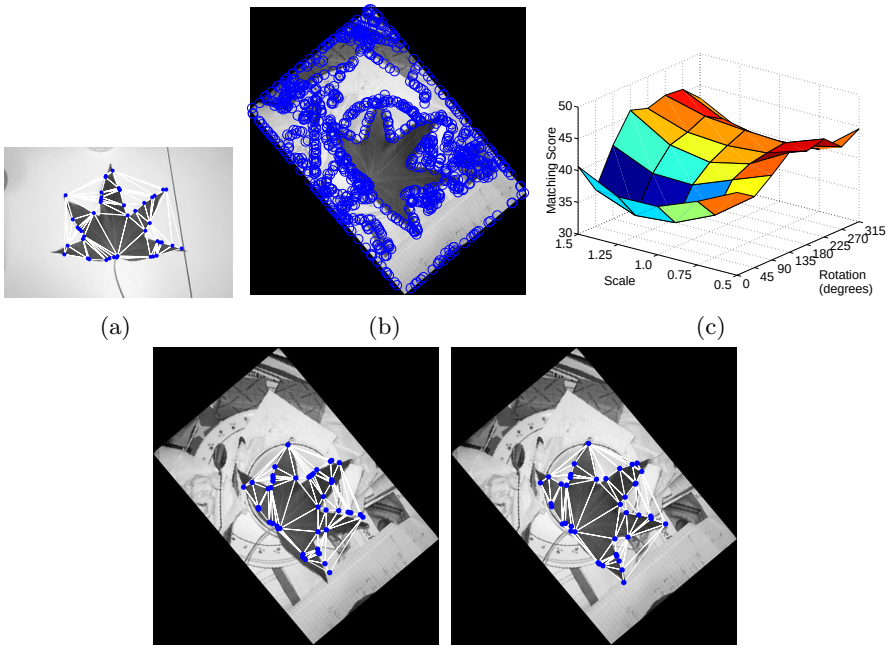
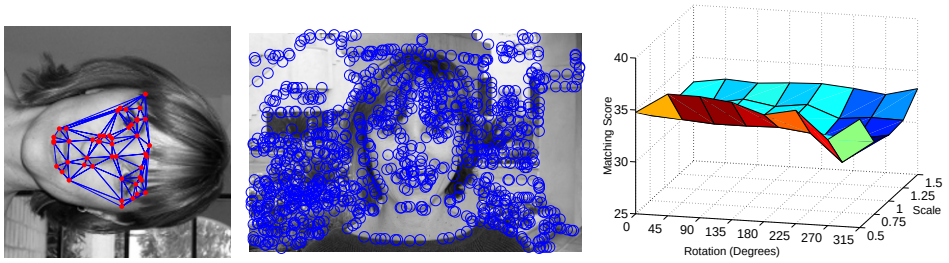


Figure 2.24: Leaf. (a): Template image and mesh; (b): Feature points on the target image; (c): Matching scores for different scales and rotations; (d): SC-QP matching in the largest trust region; (e): Final matching result. The leaf image ©Caltech, 2006, by permission.

Table 2.3: Object Detection using Caltech Face, Carback and leaf Database

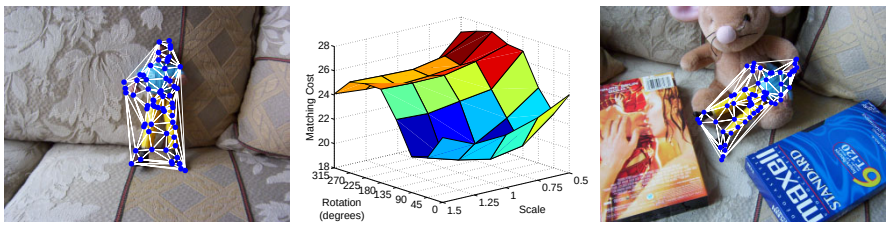
Database	Faces	Car-backs	Leaves
Success Rate	98.1% (1 template)	82.4% (1 template)	97.8% (6 templates)



(a) (b) (c)

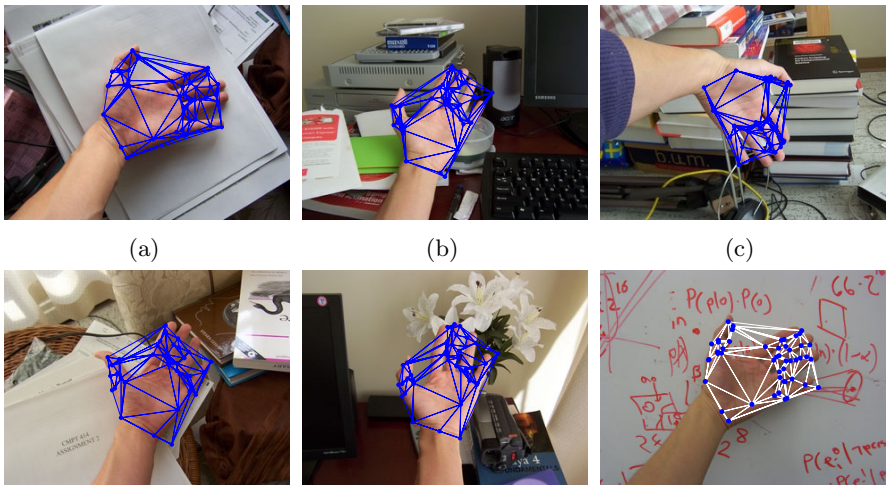
(d) (e)

Figure 2.25: Face. (a): Template image and mesh; (b): Feature points on the target image; (c): Matching scores for different scales and rotations; (d): SC-QP matching in the largest trust region; (e): Final matching result. The face image ©Caltech, 2006, by permission.



(a) (b) (c)

Figure 2.26: Toy. (a): Template image and mesh; (b): Matching scores for different scale and rotations. (c): SC-QP matching result.



(a) (b) (c)

(d) (e) (f)

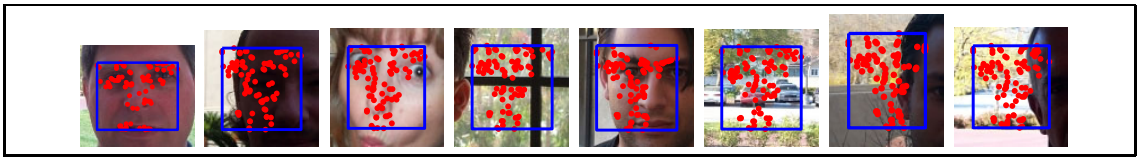
Figure 2.27: Hand. (a): Template image and mesh; (b)-(f): Hand localization results with SC-QP.

2.12 Summary

In this chapter we propose a novel successive convexification method for solving consistent labeling problems with convex regularization terms. Because of highly non-convexity of the label cost term, these problems are difficult to solve using traditional methods. The proposed scheme approximates the label cost surfaces with their lower convex hulls and constructs convex programming based on small number of basis labels. Basis labels correspond to the $(n+1)$-D lower convex hull vertices and usually have much smaller number than the candidate labels. The successive relaxation scheme then shrinks the trust regions systematically and converts the labeling problem into a sequence of much easier convex programming problems. Because of the construction procedure, the size of the convex programming problem only depends on the shape of the lower convex hulls of the labeling costs and is in a great degree decoupled with the number of labels. This enables the proposed scheme to be applied to very large label set problems. The convex labeling method is quite general and can be applied to any convex regularization problems. For L_1 norm regularization term problems, we have a linear programming relaxation, which can be solved efficiently. For L_2 regularization term problems, the non-convex problem can be relaxed into convex quadratic programming and solved using successive convexification.

(a) Template (b) Example 1 (c) Example 2 (d) Example 3 (e) Example 4 (f) Example 5

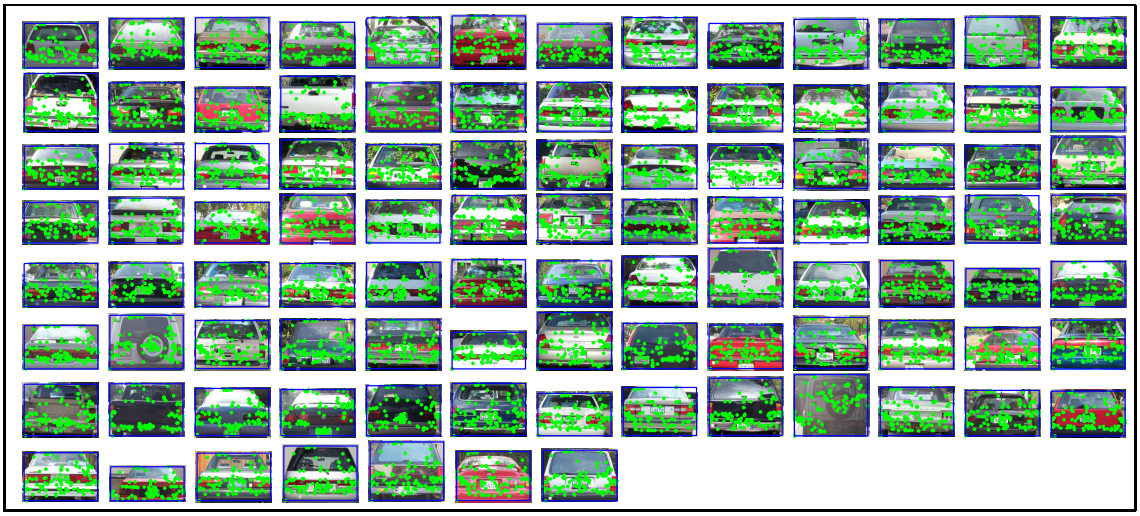
(g) Samples of correct matches



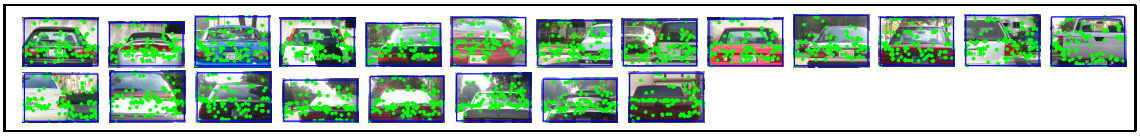
(h) Incorrect matches (8 faces).

Figure 2.28: Color face template matching. (a): Template; (b-f): Five example targets — bounding box for match is shown; More correct matches are shown in; (g): detail block near bounding box automatically cropped from the target image is shown; (h): detail block near bounding box automatically cropped from wrong matching result. The face dataset ©Caltech, 2006, by permission.

(a) Template (b) Example 1 (c) Example 2 (d) Example 3 (e) Example 4 (f) Example 5

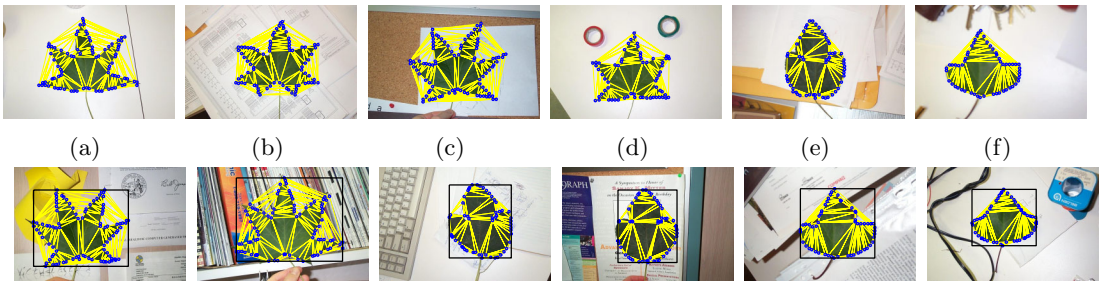


(g) Correct matches (98 of 119 cars)



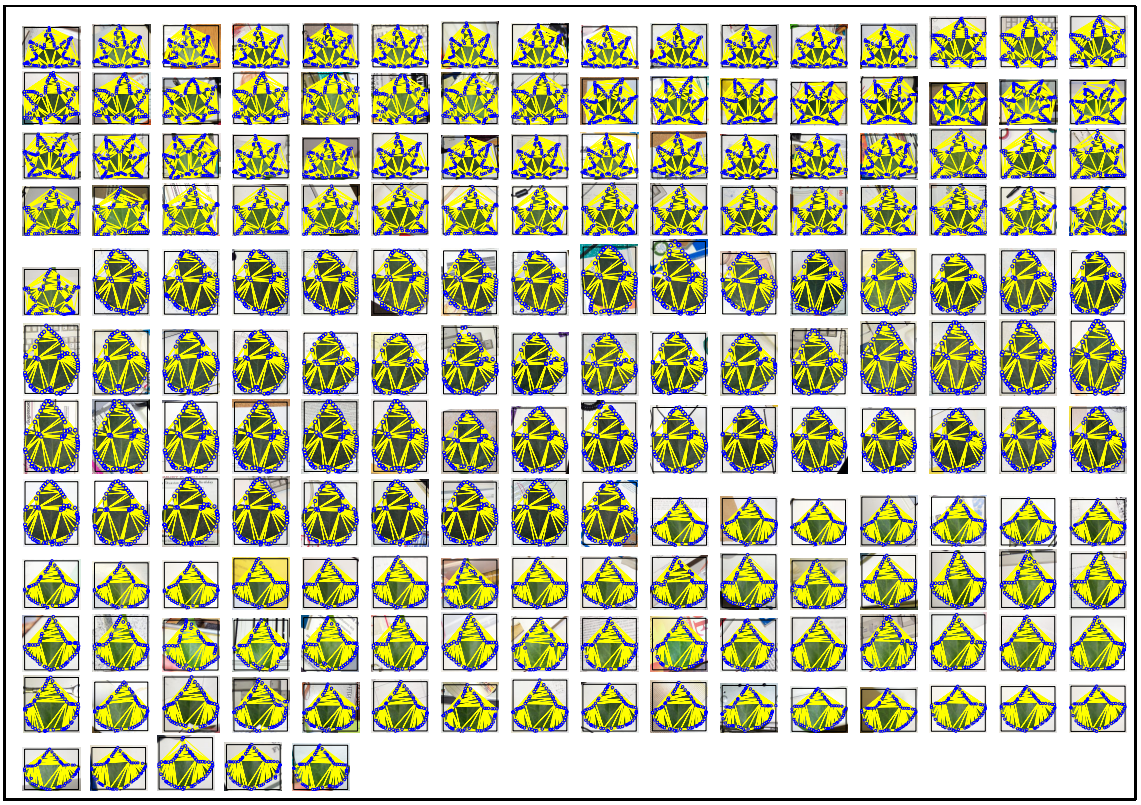
(h) Incorrect matches (21 cars).

Figure 2.29: Matching edge map by distance transform. In (g, h), blocks shown are targets automatically cropped from the target images. The carback dataset ©Caltech, 2006, by permission.

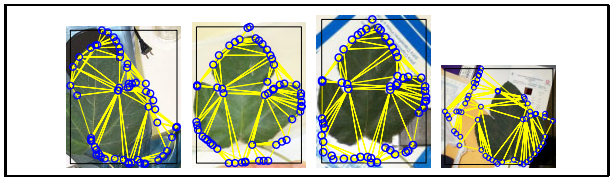


(a) (b) (c) (d) (e) (f)

(g) Example 1 (h) Example 2 (i) Example 3 (j) Example 4 (k) Example 5 (l) Example 6



(m) Correct matches (181 of 185 leaves)



(n) Incorrect matches

Figure 2.30: Matching result of leaves. (a, b, c, d, e, f): Templates; (g, h, i, j, k, l): Matching examples; (m): Correct matching blocks automatically cropped from the target images; (n): Incorrect matching blocks cropped from target images. The leaf dataset ©Caltech, 2006, by permission.

We applied the proposed convex labeling method to image matching and object localization problems. Experiments confirm that the proposed scheme is much more robust than greedy schemes. It also has advantage of other robust matching schemes such as BP because of its efficiency for very large label set problem.

Chapter 3

Optimizing Motion and Shape Estimation

Large scale motion estimation is a challenging problem for which traditional local searching schemes such as optical flow methods do not yield satisfactory results. In this chapter, we present a new method [14] to optimize large scale motion estimation from two images based on the successive convexification scheme. The proposed scheme can also estimate the occlusion map simultaneously. This chapter extends the basic successive convexification scheme with occlusion and outlier inference as shown in Fig. 3.1. Based on the correspondence obtained from motion estimation, we can further reconstruct 3-D shape with a hierarchical method.

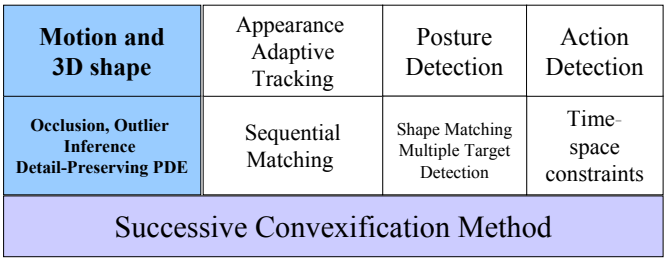


Figure 3.1: Successive convexification and extension for motion estimation.

3.1 Successive Convexification for Motion Estimation

Motion estimation is inherently a labeling problem and can be solved with the proposed successive convexification scheme. Here, we extend the formulation so that occlusion can be estimated simultaneously with motion field. The inclusion of occlusion inference makes the formulation more complex. But interestingly, the new problem shares many similar properties with the basic consistent labeling formulation in the previous chapter. The successive convexification scheme can still be applied. We present the properties and justify the extension of the successive convexification method into this problem.

To include occlusion inference in motion estimation, we use the following strategy [14]. We introduce a variable to indicate whether a site in a reference image is occluded in a target image. We choose a constant cost for labeling a pixel in reference image as occluded. A proper constant would be selected larger than the cost of a true match and smaller than that for the wrong matches. If a site is recognized as occluded, the site is assigned the constant cost and removed from the smoothing term. Different from the definition for labels in the previous chapter, here the labels are motion vectors and occlusion flags. We introduce a new variable $\pi_{\mathbf{s}}$ to indicate whether site $\mathbf{s}$ is occluded in target image. If occlusion occurs, $\pi_{\mathbf{s}} = 1$ and otherwise $\pi_{\mathbf{s}} = 0$. The labeling problem can be written as

$$\min_{\mathbf{v},\pi} \left\{ \sum_{\mathbf{s}\in S} [c_{\mathbf{s},\mathbf{v_s}}(1-\pi_{\mathbf{s}}) + c_o\pi_{\mathbf{s}}] + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\mathbf{v_p}(1-\pi_{\mathbf{p}}) - \mathbf{v_q}(1-\pi_{\mathbf{q}})|| + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \eta_{\mathbf{p},\mathbf{q}}|\pi_{\mathbf{p}} - \pi_{\mathbf{q}}| \right\}$$

where $\mathbf{v_s}$ is the motion vector at site $\mathbf{s}$; $c_{\mathbf{s},\mathbf{v_s}}$ is the cost of assigning site $\mathbf{s}$ with motion vector $\mathbf{v_s}$; c_o is a constant cost for labeling one site as occluded; $\lambda_{\mathbf{p},\mathbf{q}}$ and $\eta_{\mathbf{p},\mathbf{q}}$ determine the weight of the motion and occlusion regularization terms respectively; other symbols share the same definitions as previous simpler labeling formulation. The regularization terms for motion vectors and occlusion labels both use L_1 norms.

This nonlinear optimization problem can also be relaxed into a linear program. Following similar methods in the previous chapter, we linearize the labeling cost term with a linear combination of the costs at basis labels and the regularization terms by the linear

programming trick of introducing auxiliary variables. The linear programming relaxation is

$$\min \left\{ \sum_{\mathbf{s}\in S} [\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c_{\mathbf{s},\mathbf{j}}\, \xi_{\mathbf{s},\mathbf{j}} + c_o \pi_\mathbf{s}] + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}} \sum_{m=1}^{2} (f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}) \right. \tag{3.1}$$
$$\left. + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \eta_{\mathbf{p},\mathbf{q}} (\pi^+_{\mathbf{p},\mathbf{q}} + \pi^-_{\mathbf{p},\mathbf{q}}) \right\}$$

with constraints:

$$\begin{aligned}
\sum_{\mathbf{j}\in\mathcal{B}_s} \xi_{\mathbf{s},\mathbf{j}} + \pi_\mathbf{s} &= 1,\ \forall \mathbf{s} \in S \\
\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}} \phi_m(\mathbf{j}) &= f_{\mathbf{s},m},\ \forall \mathbf{s} \in S,\ m = 1,2 \\
f_{\mathbf{p},m} - f_{\mathbf{q},m} &= f^+_{\mathbf{p},\mathbf{q},m} - f^-_{\mathbf{p},\mathbf{q},m}\,, \\
\pi_\mathbf{p} - \pi_\mathbf{q} &= \pi^+_{\mathbf{p},\mathbf{q},m} - \pi^-_{\mathbf{p},\mathbf{q},m}, \forall\ \{p,q\} \in \mathcal{N},\ m = 1,2 \\
\xi_{\mathbf{s},\mathbf{j}}, \pi_\mathbf{s} &\geq 0, \quad f^+_{\mathbf{p},\mathbf{q},m}, f^-_{\mathbf{p},\mathbf{q},m}, \pi^+_{\mathbf{p},\mathbf{q},m}, \pi^-_{\mathbf{p},\mathbf{q},m} \geq 0
\end{aligned}$$

Here, $\phi_m(\mathbf{j})$ is similarly defined as extracting the mth element of vector $\mathbf{j}$; $\mathcal{B}_\mathbf{s}$ is the basis label set for site $\mathbf{s}$. Note that vector $(f_{\mathbf{s},1}, f_{\mathbf{s},2})$ is not the motion vector $\mathbf{v}_\mathbf{s}$. In fact, it equals $\mathbf{v}_\mathbf{s}(1-\pi_\mathbf{s})$ which is also denoted as the projected motion vector [14]. Ideally, $(f_{\mathbf{s},1}, f_{\mathbf{s},2})$ equals motion vector at the non-occlusion regions and vanishes at occlusion regions. Another difference of this linear program with the basic one is that $f_{\mathbf{s},m}$ can take negative values.

This linear programming relaxation has similar properties with the previous simpler formulation. It is also closely correlated with the continuous extension of the nonlinear problem similarly defined as the previous basic one. The only difference is that we relax the newly introduced occlusion variables into float numbers in 0 – 1.

Property 3.1*: If cost function $c_{\mathbf{s},\mathbf{t}}$ over $\mathbf{t}$ is convex for each site $\mathbf{s}$, and the basis set $\mathcal{B}_\mathbf{s}$ contains all the motion candidates for each site $\mathbf{s}$, the linear programming exactly solves the continuous extension of the original problem.*

Proof: We need to show that the LP optimal solution is also the optimal solution of the continuous extension of the original problem. Assume that the LP optimal solution has configuration of $\{\xi^*, \pi^*\}$. Since $c_{\mathbf{s},\mathbf{t}}$ over $\mathbf{t}$ is convex for each site $\mathbf{s}$,

$$\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c_{\mathbf{s},\mathbf{j}} \cdot \xi^*_{\mathbf{s},\mathbf{j}} = \sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} (1-\pi^*_\mathbf{s}) c_{\mathbf{s},\mathbf{j}} \cdot \eta^*_{\mathbf{s},\mathbf{j}} \geq (1-\pi^*_\mathbf{s}) c_{\mathbf{s},\mathbf{v}^*_\mathbf{s}}$$

where $\eta^*_{\mathbf{s},\mathbf{j}} = \xi^*_{\mathbf{s},\mathbf{j}}/(1-\pi^*_{\mathbf{s}})$ if $\pi^*_{\mathbf{s}} < 1$, and otherwise any non-negative numbers that complies with $\sum_{\mathbf{j}\in\mathcal{B}_{\mathbf{s}}} \eta^*_{\mathbf{s},\mathbf{j}} = 1$; we define $\mathbf{v}^*_{\mathbf{s}} = \sum_{\mathbf{j}\in\mathcal{B}_{\mathbf{s}}} \eta^*_{\mathbf{s},\mathbf{j}} \cdot \mathbf{j}$. Apparently,

$$\sum_{\mathbf{j}\in\mathcal{B}_{\mathbf{s}}} \xi^*_{\mathbf{s},\mathbf{j}} \cdot \phi_x(\mathbf{j}) = (1-\pi^*_{\mathbf{s}})\phi_x(\mathbf{v}^*_{\mathbf{s}}), \sum_{\mathbf{j}\in\mathcal{B}_{\mathbf{s}}} \xi^*_{\mathbf{s},\mathbf{j}} \cdot \phi_y(\mathbf{j}) = (1-\pi^*_{\mathbf{s}})\phi_y(\mathbf{v}^*_{\mathbf{s}}),$$

It is not difficult to show that when the linear program is optimized we have

$$\sum_{m=1}^{2}(f^+_{\mathbf{p},\mathbf{q},m} + f^-_{\mathbf{p},\mathbf{q},m}) = ||\mathbf{v}^*_{\mathbf{p}}(1-\pi^*_{\mathbf{p}}) - \mathbf{v}^*_{\mathbf{q}}(1-\pi^*_{\mathbf{q}})||, \pi^+_{\mathbf{p},\mathbf{q}} + \pi^-_{\mathbf{p},\mathbf{q}} = |\pi^*_{\mathbf{p}} - \pi^*_{\mathbf{q}}|$$

Based on the definition of continuous extension, $\{\mathbf{v}^*_{\mathbf{s}}, \pi^*_{\mathbf{s}}\}$ is a feasible solution to the nonlinear problem. Therefore,

$$\begin{aligned}\min(\text{LP}) \quad &\geq \quad \sum_{s\in S}\left[c_{\mathbf{s},\mathbf{v}^*_{\mathbf{s}}}(1-\pi^*_{\mathbf{s}}) + c_o \cdot \pi^*_{\mathbf{s}}\right] + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \{\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{v}_{\mathbf{p}}(1-\pi^*_{\mathbf{p}}) \\ &\qquad -\mathbf{v}_{\mathbf{q}}(1-\pi^*_{\mathbf{q}})|| + \ \mu_{\mathbf{p},\mathbf{q}}|\pi^*_{\mathbf{p}} - \pi^*_{\mathbf{q}}|\} \\ &\geq \quad \min(\text{Nonlinear Problem})\end{aligned}$$

On the other hand, we assume that $\{\mathbf{v}_{\mathbf{s}}, \pi_{\mathbf{s}}\}$ is a feasible solution of the continuous extension of the nonlinear optimization problem. For each site, we can solve a minimization problem like the one in Property 2.1 to get $\eta_{\mathbf{s},\mathbf{j}}$ and then scale $\eta_{\mathbf{s},\mathbf{j}}$ down by $(1-\ \pi_{\mathbf{s}})$ and get $\xi_{\mathbf{s},\mathbf{j}}$. Other variables in LP can be set accordingly. This solution is feasible to the linear program and has no greater objective function than that of the nonlinear problem. Because this follows for any feasible solution of the nonlinear problem, we have

$$\min(\text{LP}) \leq \min(\text{Nonlinear Problem}).$$

Therefore

$$\begin{aligned}&\sum_{s\in S}\left[c_{\mathbf{s},\mathbf{v}^*_{\mathbf{s}}}(1-\pi^*_{\mathbf{s}}) + c_o \cdot \pi^*_{\mathbf{s}}\right] + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \{\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{v}_{\mathbf{p}}(1-\pi^*_{\mathbf{p}}) \\ &-\mathbf{v}_{\mathbf{q}}(1-\pi^*_{\mathbf{q}})|| + \ \mu_{\mathbf{p},\mathbf{q}}|\pi^*_{\mathbf{p}} - \pi^*_{\mathbf{q}}|\} \\ = \quad &\min(\text{Nonlinear Problem})\end{aligned}$$

The property follows.

The following property shows that the proposed linear programming method corresponds to reformulating the original non-convex problem into an approximated convex programming problem.

Property 3.2*: If basis* $\mathcal{B}_\mathbf{s}$ *contains all the motion vector candidates for each site* $\mathbf{s}$*, the linear program optimizes a reformulated non-linear optimization problem with* $c_{\mathbf{s},\mathbf{t}}$ *approximated by its lower convex hull for each site* $\mathbf{s}$*.*

The proof is similar to property 3.1 by replacing $c_{\mathbf{s},\mathbf{t}}$ with lower convex hull surfaces.

Property 3.3: The most compact basis $\mathcal{B}_\mathbf{s}$ contains the motion vectors corresponding to the lower convex hull of $(\phi_1(\mathbf{t}), \phi_2(\mathbf{t}), c_{\mathbf{s},\mathbf{t}})$ for each site $\mathbf{s}$.

Therefore we may construct the linear program with only the labels corresponding to the lower convex hull vertices for each matching surface without changing the solution of the LP. This will usually remove many variables and greatly speeds up the algorithm.

Property 3.4: The basis feasible solutions of the LP contains at most three non-zero ξ for each site.

Proof: We follow the proof of Property 2.5. Note that we can always replace $f_{\mathbf{s},\mathbf{1}}$ and $f_{\mathbf{s},\mathbf{2}}$ with the difference of nonnegative auxiliary variables. Then, for each site $\mathbf{s}$, variable $\xi_{\mathbf{s},\mathbf{j}}$ introduces a column $[0..., 1, \phi_1(\mathbf{j}), \phi_2(\mathbf{j}), 0...]^T$ in the constraint coefficient matrix. The submatrix generated by these columns for a single site has a rank of at most 3. The property follows.

We can also formulate an approximation algorithm by converting LP solutions to discrete labels. Such approximation algorithm is useful in estimating an upper bound in the successive convexification algorithm. As the basic formulation, we can convert the LP result into discrete label space by choosing the largest coefficient from $\xi_{\mathbf{s},\mathbf{j}}$ and $\pi_\mathbf{s}$ for each site and set it 1 and others 0.

Property 3.5: *For each site, if we round the largest* $\xi_{\mathbf{s},\mathbf{j}}$ *and* $\pi_\mathbf{s}$ *to 1 and the rest to zero, the approximation algorithm has a upper bound* $4E_{opt} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}[\lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{f_p}-\hat{\mathbf{f}}_\mathbf{p}|| + ||\mathbf{f_q}-\hat{\mathbf{f}}_\mathbf{q}||) + \mu_{\mathbf{p},\mathbf{q}}(|\pi_\mathbf{p} - \hat{\pi}_\mathbf{p}| + |\pi_\mathbf{q} - \hat{\pi}_\mathbf{q}|]$*, where* $\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \hat{\xi}_{\mathbf{s},\mathbf{j}} \cdot \mathbf{j} = \hat{\mathbf{f}}_\mathbf{p}$ and $\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} \xi_{\mathbf{s},\mathbf{j}} \cdot \mathbf{j} = \mathbf{f_p}$, $\hat{\xi}_{\mathbf{s},\mathbf{j}}$ and $\hat{\pi}_\mathbf{s}$ are rounded $\xi_{\mathbf{s},\mathbf{j}}$ and $\pi_\mathbf{s}$ respectively; E_{opt} is the optimum of the discrete labeling problem.

Proof: The rounded solution is feasible to the nonlinear discrete labeling problem. A direct conclusion from Property 3.4 is that for each site $\mathbf{s}$, $\max(\xi_{\mathbf{s},\mathbf{j}}, \pi_\mathbf{s}) \geq 1/4$ (else the sum would be necessarily less than 1). Therefore, rounding ξ to $\hat{\xi}$, $\pi_\mathbf{s}$ to $\hat{\pi}_\mathbf{s}$,

$$\sum_{\mathbf{s}\in S}[\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c_{\mathbf{s},\mathbf{j}}\,\hat{\xi}_{\mathbf{s},\mathbf{j}} + c_o\hat{\pi}_\mathbf{s}] \leq 4\sum_{\mathbf{s}\in S}[\sum_{\mathbf{j}\in\mathcal{B}_\mathbf{s}} c_{\mathbf{s},\mathbf{j}}\,\xi_{\mathbf{s},\mathbf{j}} + c_o\pi_\mathbf{s}]$$

Based on triangle inequality,

$$\begin{aligned}\sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}||\hat{\mathbf{f}}_{\mathbf{p}}-\hat{\mathbf{f}}_{\mathbf{q}}|| &\leq \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \lambda_{\mathbf{p},\mathbf{q}}(||\hat{\mathbf{f}}_{\mathbf{p}}-\mathbf{f}_{\mathbf{p}}|| + ||\mathbf{f}_{\mathbf{p}}-\mathbf{f}_{\mathbf{q}}|| + ||\hat{\mathbf{f}}_{\mathbf{q}}-\mathbf{f}_{\mathbf{q}}||)\\ \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \mu_{\mathbf{p},\mathbf{q}}|\hat{\pi}_{\mathbf{p}}-\hat{\pi}_{\mathbf{q}}| &\leq \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} \eta_{\mathbf{p},\mathbf{q}}(|\hat{\pi}_{\mathbf{p}}-\pi_{\mathbf{p}}| + |\pi_{\mathbf{p}}-\pi_{\mathbf{q}}| + |\hat{\pi}_{\mathbf{q}}-\pi_{\mathbf{q}}|)\end{aligned}$$

Considering the fact that $\sum_{\mathbf{s}\in S}[\sum_{\mathbf{j}\in\mathcal{B}_{\mathbf{s}}} c_{\mathbf{s},\mathbf{j}}\xi_{\mathbf{s},\mathbf{j}} + c_o\pi_{\mathbf{s}}] + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}(\lambda_{\mathbf{p},\mathbf{q}}||\mathbf{f}_{\mathbf{p}}-\mathbf{f}_{\mathbf{q}}|| + \mu_{\mathbf{p},\mathbf{q}}|\pi_{\mathbf{p}} - \pi_{\mathbf{q}}|) \leq E_{opt}$, the proposition follows. For practical computer vision problems, most of the ξ and π in fact approach 1 or 0 and therefore $\hat{\mathbf{f}}_{\mathbf{s}}$ and $\hat{\pi}_{\mathbf{s}}$ approach $\mathbf{f}_{\mathbf{s}}$ and $\pi_{\mathbf{s}}$ respectively.

The simple rounding process does not consider the neighboring relations. To enforce the neighboring constraint in the rounding, we can also use the following consistent rounding process: we check the discrete labels (motion candidates and occlusion) and select the label that minimizes the nonlinear objective function, given the configuration of continuous labels defined by the solution of the current stage. We define that $\mathbf{m}_{\mathbf{s}}$ and $O_{\mathbf{s}}$ are the global optimal motion (projected motion) and occlusion solution respectively, and $\mathbf{f}_{\mathbf{s}}$ and $\pi_{\mathbf{s}}$ are the continuous motion and occlusion solution of LP.

Property 3.6: The energy with consistent rounding is bounded above by
$3E_{opt} + \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}}[\lambda_{\mathbf{p},\mathbf{q}}(||\mathbf{m}_{\mathbf{p}}-\mathbf{f}_{\mathbf{p}}|| + ||\mathbf{m}_{\mathbf{q}}-\mathbf{f}_{\mathbf{q}}||) + \mu_{\mathbf{p},\mathbf{q}}(|O_{\mathbf{p}} - \pi_{\mathbf{p}}| + |O_{\mathbf{q}} - \pi_{\mathbf{q}}|)]$.

As discussed above, a single LP relaxation approximates the original problem's motion cost functions by their lower convex hulls. This process can be rough if motions for small image blocks have large ambiguity. To solve the coarse approximation problem, similar to the simpler formulation in the last chapter, we can shrink the search region for each site based on the current LP solution, and do a further search by solving a new LP problem in the smaller trust regions. The new LP in smaller trust regions reconvexifies the cost function within the focus regions only. The successive reconvexification LP scheme can thus be used to improve the motion and occlusion estimation.

In the process of successive convexification for motion estimation, the trust region for each site corresponds to a rectangular search area in the target image. Initially, the trust region for each site corresponds to the largest searching window which is usually a rectangular region centered on each site. As mentioned before, the LP solution $(f_{\mathbf{s},1}, f_{\mathbf{s},2})$ is a projected motion vector which vanishes at occlusion regions and equals the true motion in non-occlusion areas. This introduces an undesired penalty at the occlusion and non-occlusion boundaries. This penalty varies with the object's motion magnitude. For medium scale motions in the range of [-30,30] pixels in both x and y directions, this does not pose

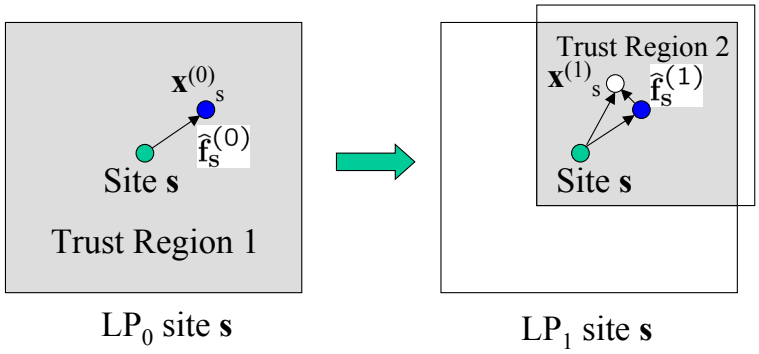


Figure 3.2: Region shrinking for successive convexification motion estimation.

a problem. If an object has a very large motion, this penalty will become quite big and the single step LP method may mis-classify the occlusion region caused by this object as non-occluded. This problem can be solved by an alternate successive relaxation scheme.

This successive convexification procedure is illustrated in Fig. 3.2. In this method, we iteratively construct a sequence of linear programs LP_n, $n = 0..N$. After solving a linear program in search region n, we store the correspondence point $\mathbf{x}_{\mathbf{s}}^{(n)} = \mathbf{x}_{\mathbf{s}}^{(n-1)} + \hat{\mathbf{f}}_{\mathbf{s}}^{(n)}$, $n = 0..N$, for each site $\mathbf{s}$, where $\hat{\mathbf{f}}_{\mathbf{s}}^{(n)}$ is the consistent rounding motion vector based on LP_n solution $\mathbf{f}_{\mathbf{s}}^{(n)}$. We set $\mathbf{x}_{\mathbf{s}}^{(-1)} = \mathbf{s}$ and LP_0 has the search region initialized to be the largest searching window for each site. We shrink the search region for each site and build convex programming in these new regions. The motion cost $c_{\mathbf{s},\mathbf{v}}$ for LP_n is rewritten as $c_{\mathbf{s},\mathbf{v}}^{(n)}$ and defined as the cost of assigning motion vector $\mathbf{v} + \mathbf{x}_{\mathbf{s}}^{(n-1)} - \mathbf{s}$ to site $\mathbf{s}$. We denote $c_{\mathbf{s},\mathbf{v}}^{(n)} = \mathcal{M}(\mathbf{s}, \mathbf{v} + \mathbf{x}_{\mathbf{s}}^{(n-1)} - \mathbf{s})$, where $\mathcal{M}(\mathbf{p}, \mathbf{u})$ is the cost of matching reference pixel $\mathbf{p}$ with target pixel $\mathbf{u} + \mathbf{p}$. For LP_0, $c_{\mathbf{s},\mathbf{v}}^{(0)}$ has the same definition as the single LP relaxation, which is the cost of assigning motion vector $\mathbf{v}$ to site $\mathbf{s}$. Since the absolute motion vectors in the trust regions will finally become small, the large motion occlusion boundary problems are solved. We still do not need to recompute the motion costs in each iteration if we always intersect each new search region with the initial one.

Algorithm 3.1 *Successive Convexification for Motion Estimation*

1. Set initial search regions for each site as the largest searching window;
   set $\mathbf{x}_{\mathbf{s}}^{(-1)} = \mathbf{s}$, $\mathbf{s} \in S$; $n = 0$;
2. Compute motion cost surface for each site and construct LP_0 based on Eq. 3.1;
3. Solve LP_0; $\mathbf{x}_{\mathbf{s}}^{(0)} = \mathbf{x}_{\mathbf{s}}^{(-1)} + \hat{\mathbf{f}}_{\mathbf{s}}^{(0)}$;

4. While (Trust Region Diameter $> d_{\min}$)
5. $n \leftarrow n+1$
6. Shrink trust regions and center new search region at $\mathbf{x}_{\mathbf{s}}^{(n-1)}$ for each site $\mathbf{s}$;
7. Re-compute lower convex hull for each site based on $c_{\mathbf{s},\mathbf{v}}^{(n)} = \mathcal{M}(s, \mathbf{v} + \mathbf{x}_{\mathbf{s}}^{(n-1)} - \mathbf{s})$;
8. Construct and solve new LP_n;
9. Consistent rounding: $\mathbf{f}_{\mathbf{s}}^{(n)} \rightarrow \hat{\mathbf{f}}_{\mathbf{s}}^{(n)}$;
10. Update correspondence points $\mathbf{x}_{\mathbf{s}}^{(n)} = \mathbf{x}_{\mathbf{s}}^{(n-1)} + \hat{\mathbf{f}}_{\mathbf{s}}^{(n)}$;
11. Output $\mathbf{f}_{\mathbf{s}}^{*}$ and $\pi_{\mathbf{s}}^{*}$, $\mathbf{s} \in S$;

3.2 Combining Successive Convex Programming and Detail Preserving PDE

In the following, we study a two-stage method to estimate a dense motion field. In the first stage, we use successive convexification method to get a pseudo-dense motion and occlusion field. In the second stage, we upgrade the motion into dense motion field based on a detail preserving PDE method. The two-step method greatly speeds up the motion estimation process. Because of the robust motion estimation at the first stage, local search using PDE method will drive the dense motion estimation approaching the global optimum.

3.2.1 Pseudo-dense Motion Estimation

We randomly select ρ of the total edge pixels, where ρ is in $(0, 1]$. A typical ρ is from 0.1 to 0.5. We also randomly select κ of the non-edge pixels, with κ in $(0, 1]$ and $\kappa \leq \rho$. The non-edge pixels usually have less reliability in the local motion searching but can be used to improve the uniformity of the sampling scheme and the final dense motion field estimation. Assume S is the set of chosen pixels for pseudo-dense motion estimation. We Delaunay-triangulate the point set S. The Delaunay graph edges define the neighbor relation $\mathcal{N}$. The motion basis set $\mathcal{B}_{x,y}$ for site (x, y) contains coordinates of the vertices for lower convex hull of $\{x, y, c((x, y), (u, v))\}$, where (u, v) are all motion vectors in a search region. We confine the motion searching in a rectangular window centered at site (x, y) in the reference image which corresponds to the motion candidate set $W = [-\Delta_x, \Delta_x] \times [-\Delta_y, \Delta_y]$. Here, we simply take the cost of assigning motion (u, v) to site (x, y) to be the *normalized absolute*

difference between reference and matching blocks:

$$c(x,y,u,v) = \frac{\sum_{(x',y')\in R_{x,y}} |I_r(x',y') - I_m(x'+u,y'+v)|}{(2\tau+1)^2 \cdot (\sigma_r^2+\varepsilon)^{1/2}(\sigma_m^2+\varepsilon)^{1/2}}$$

where I_r and I_t are the normalized reference and target gray level images, respectively; $R_{x,y} = [x-\tau, x+\tau] \times [y-\tau, y+\tau]$, where τ is 1..4; σ_r^2 and σ_m^2 are the non-deviation estimations of the variance of the image blocks; ε is a small positive number to avoid zero denominator.

In our scheme, we would like the smoothing terms λ and μ adapted to the distance of neighboring sites: $\lambda_{(x,y),(x',y')} = \lambda_0 w(d_{(x,y),(x',y')})$ and $\mu_{(x,y),(x',y')} = \mu_0 w(d_{(x,y),(x',y')})$ where $d_{(x,y),(x',y')} = ((x-x')^2 + (y-y')^2)^{1/2}$ and $w(\cdot)$ is a decreasing function in the domain $[0,1]$, with λ_0 and μ_0 two constant coefficients. Here, $w(\cdot)$ is selected as a simple staircase function equal to 0 if the distance between two neighbors is greater than some threshold and is otherwise 1.

Based on the proposed successive convexification method, we obtain a pseudo-dense estimation for the motion in the x and y directions, as well as the occlusion map.

3.2.2 Dense Refinement

The pseudo-dense motion needs further refinement to derive a dense estimation at a resolution equal to that of the original images. The occlusion map can easily be extended to a dense map by a simple interpolation scheme. In this section we give a scheme to refine the x and y motion field into a dense motion field. We use the interpolated pseudo-dense x-, y-motion field as the initial motion, and apply a PDE-based method to refine the estimation in a force field determined by the labeling cost function $c(x,y,u,v)$.

To begin with, we interpolate the pseudo-dense x-motion u_{x_i,y_i} so as to generate a dense surface $h(x,y)$. A straightforward approach for achieving this is to solve a constrained Laplace's equation:

$$\begin{cases} \nabla^2 h = 0 \\ h(x_i,y_i,t) \equiv u_{x_i,y_i},\ t>0, i=1...M \end{cases}$$

for h. To do so, we iterate using an artificial time variable t, and constrain a solution at any time to be fixed at points (x_i,y_i). Dense x-motion $h(x,y)$ is analogous to the steady state for isotropic heat transmission with constant temperatures at specific points. y-motion is interpolated with a similar scheme and the dense y-motion is denoted $v(x,y)$.

We further formulate the following variational problem to derive smooth motion fields p and q (x and y motions) from the initial interpolated h and v:

$$\{\widehat{p},\widehat{q}\} = \min_{p,q} \mathcal{J} = \int_x \int_y [\eta \cdot c(x,y,p,q) + \frac{(1-\alpha(x,y))}{2}\|\nabla p(x,y)\|^2 + \frac{(1-\beta(x,y))}{2}\|\nabla q(x,y)\|^2 + \frac{\alpha(x,y)}{2}\|\nabla p(x,y)\| + \frac{\beta(x,y)}{2}\|\nabla q(x,y)\|]dxdy, \tag{3.2}$$

with constraints

$$\frac{p_y}{\sqrt{p_x^2+p_y^2}} = \sin[\theta(x,y)], \frac{p_x}{\sqrt{p_x^2+p_y^2}} = \cos[\theta(x,y)], \frac{q_y}{\sqrt{q_x^2+q_y^2}} = \sin[\varphi(x,y)], \frac{q_x}{\sqrt{q_x^2+q_y^2}} = \cos[\varphi(x,y)], \alpha = f^{(x)}(\|\nabla h^\sigma(x,y)\|), \beta = f^{(y)}(\|\nabla v^\sigma(x,y)\|), \tag{3.3}$$

where the $\{\sin, \cos\}$ constraints signify that p and q are surfaces. We preserve smooth motions by letting function $\alpha(x,y)$ or $\beta(x,y)$ be small if the initial interpolating motion h or v is smooth. In that case, we would like the second and third terms in $\mathcal{J}$ to dominate and cost function to guide further smoothing. However, for non-smooth motions we would like the fourth and fifth, curvature-producing, terms to be more important and therefore $\alpha(x,y)$ or $\beta(x,y)$ should be larger. Coefficient η is a constant that controls the influence of the energy term based on the consistency function. $f^{(\cdot)}(x) = \max((\frac{x-x_{\min}}{x_{\max}-x_{\min}})^{0.2}, 0.95)$, where $x_{\min}$ and $x_{\max}$ are the minimum and maximum gradients in an image.

So we use the initial interpolating motion h and v to set a general smoothness level by defining positive monotonic-increasing functions $f^{(x)}(\cdot)$ and $f^{(y)}(\cdot)$ both with range $[0,1]$, so that α and β go from low to high as smoothness decreases. To simplify the variational equation below, we further assume that the motion field can be represented as piecewise planar surfaces. Thus the gradients of the x-motion field and y-motion field are piecewise constant. Based on the definitions of α and β, we have $\partial\alpha/\partial x \simeq 0, \partial\alpha/\partial y \simeq 0, \partial\beta/\partial x \simeq 0, \partial\beta/\partial y \simeq 0$ almost everywhere. A simplifying assumption, then, is that the partial derivatives of the α and β vanish. Based on this assumption, the resulting PDE below becomes much simpler, crucially for an iterative algorithm, at the expense of slightly sacrificing the edge preserving property.

We also use the initial normal vectors ∇h and ∇v to set the normal direction for a solution, ∇p and ∇q, by constraining p and q via the first four conditions in (3.3). To fix the normal direction, we take $\theta(x,y) = \arctan(h_y^\sigma/h_x^\sigma)$, with h^σ the initial dense x-motion h, Gaussian-smoothed at scale σ, and similarly $\varphi(x,y) = \arctan(v_y^\sigma/v_x^\sigma)$.

The Euler-Lagrange equations for Eq. (3.2) are given by the variational derivatives

$\delta\mathcal{J}/\delta p = 0, \delta\mathcal{J}/\delta q = 0$, and applying the vanishing assumptions for the partial derivatives of α and β we have

$$\begin{aligned} \eta c_p(x,y,p,q) - (1-\alpha)(p_{xx}+p_{yy}) - \alpha\frac{p_{xx}p_y^2+p_{yy}p_x^2-2p_{xy}p_xp_y}{2(p_x^2+p_y^2)^{3/2}} = 0, \\ \eta c_q(x,y,p,q) - (1-\beta)(q_{xx}+q_{yy}) - \beta\frac{q_{xx}q_y^2+q_{yy}q_x^2-2q_{xy}q_xq_y}{2(q_x^2+q_y^2)^{3/2}} = 0 \end{aligned} \tag{3.4}$$

The third term in both equations— the curvature for equi-value contour curves for fields p and q— smoothes only along contour tangents (when the terms contribute enough, i.e., at edges), not isotropically like the second term.

Substituting Eq. (3.3) into (3.4), by introducing an artificial time variable t and adding a small positive value ε to avoid a zero denominator, the partial differential equation solution of the system can be written

$$\begin{aligned} p_t = p_{xx}(1-\alpha+\alpha\frac{\sin^2\theta}{2(p_x^2+p_y^2)^{1/2}+\varepsilon}) + p_{yy}(1-\alpha+\alpha\frac{\cos^2\theta}{2(p_x^2+p_y^2)^{1/2}+\varepsilon}) \\ -p_{xy}\frac{\alpha\sin\theta\cos\theta}{(p_x^2+p_y^2)^{1/2}+\varepsilon} - \eta c_p(x,y,p,q), \\ q_t = q_{xx}(1-\beta+\beta\frac{\sin^2\varphi}{2(q_x^2+q_y^2)^{1/2}+\varepsilon}) + q_{yy}(1-\beta+\beta\frac{\cos^2\varphi}{2(q_x^2+q_y^2)^{1/2}+\varepsilon}) \\ -q_{xy}\frac{\beta\sin\varphi\cos\varphi}{(q_x^2+q_y^2)^{1/2}+\varepsilon} - \eta c_q(x,y,p,q) \end{aligned}$$

In summary, the x- and y-motion are initialized to h and v respectively, the dense interpolation of the sparse motion. As time t increases, the above PDE approaches a standard heat equation with a penalty term given by the motion cost function in smooth motion regions. The diffusion method approaches a mean curvature motion solution at rapidly changing areas such as edges or corners. A finite difference method is used in the discretization of the PDEs. The coupled PDEs are iterated alternatively between the x- and y-motion fields until the result converges.

3.3 Shape from Motion

Based on the proposed successive convexification motion estimation scheme, we can further reconstruct 3-D surface shapes from two uncalibrated images. We assume that (1): the camera principle point is at the center of an image; (2): x and y axes are orthogonal and we preset the focal length to be some reasonable value. Using the image motion vectors

estimated from the successive convexification motion estimation scheme we can obtain the camera's fundamental matrix F. We assume K is the camera intrinsic matrix obtained based on the assumption. Then, the essential matrix $E = K^T F K = [\mathbf{t}]_\times R$. We need to decompose E to obtain translate $\mathbf{t}$ and rotation matrix R, which can be done based on Tsai's method [82]. This decomposition is not unique. The rotation is ambiguous up to π and the translation up to a scale factor. But this usually does not pose a problem in our application, since rotation in our problem must be less than π. Scaling is also acceptable in most applications. Based on the reconstructed camera matrices, we may further reconstruct 3-D surfaces.

3.3.1 Coarse to Fine Surface Reconstruction

We first reconstruct a 3-D non-regular mesh embedded in a discretized space. This problem can be solved based on a max-flow scheme. In contrast to most voxel-based methods, instead of using all the pixels for 3-D surface reconstruction we reconstruct only the 3-D position of "important points" on the object. These important points are much denser than the sparse reconstruction while at the same time sparser than a traditional dense 3-D reconstruction method. We use *edge points* as the important points. This approach yields results as good as a brute-force dense reconstruction method since texture areas contain most of the information for 3-D reconstruction. In our scheme we use a graph construction method which guarantees the single cut condition, without using infinite edges as in [4]. We study how to choose the capacities for vertical edges such that the simple cut condition is guaranteed. The greatly reduced number of edges gives this method an advantage in both time and space efficiency. After the first reconstruction step, we obtain a *non-regular mesh*, which is ready for use in many applications such as recognition tasks. But the reconstruction may still not be good enough for some modeling applications because of the inherent staircase artifacts from the max-flow method. Therefore we further develop an adaptive PDE scheme to upgrade the 3-D mesh into a smooth surface. The proposed hierarchical method is found to be faster than a pure min-cut based method and also has greatly diminished staircase effects. Since a coarse 3-D mesh is first constructed based on a global optimization algorithm — the max-flow method — the method is much more robust than a pure PDE or variational approach.

Space Partition

We base a space partition on one reference camera. The volume of interest is defined by the intersection of the camera cone and two bounding planes.

We consider the reference camera in a given world coordinate system. For any 3-D point $\mathbf{P}$ in the homogeneous representation and its corresponding image pixel $\mathbf{p}$, assuming the optical center of the reference camera is $\mathbf{C}$, also in homogeneous coordinates, the ray starting from $\mathbf{C}$ and passing through $\mathbf{p}$ obeys $\alpha\mathbf{C} + \mathbf{P}_0$, where $\mathbf{P}_0 = \mathbf{M}^+\mathbf{p}$, with $\mathbf{M}^+$ the pseudo-inverse of camera matrix $\mathbf{M}$ and α a parameter specifying the position of the point on the ray. We introduce two parallel bounding planes denoted $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$. The intersection of the ray and plane $\mathbf{\Pi}_1$ satisfies $\mathbf{\Pi}_1^T(\alpha_1\mathbf{C}+\mathbf{P}_0) = 0$; therefore we have $\alpha_1 = -\mathbf{\Pi}_1^T\mathbf{P}_0/\mathbf{\Pi}_1^T\mathbf{C}$. Similarly, $\alpha_2 = -\mathbf{\Pi}_2^T\mathbf{P}_0/\mathbf{\Pi}_2^T\mathbf{C}$. Then we have the following observation.

Observation 3.1. For any pixel $\mathbf{p}$ on the reference image, the intersections of the ray passing through the pixels are denoted by α_1, α_2. The point with $\alpha_1\beta + \alpha_2(1-\beta)$ on the ray falls on the plane $\eta\mathbf{\Pi}_1 + \zeta\mathbf{\Pi}_2$ parallel to the two bounding planes, with $\eta = \frac{\mathbf{\Pi}_2^T\mathbf{C}}{1-\beta}$ and $\zeta = \frac{\mathbf{\Pi}_1^T\mathbf{C}}{\beta}$.

Proof: Since $\mathbf{\Pi}_1^T(\alpha_1\mathbf{C} + \mathbf{P}_0) = 0$ and $\mathbf{\Pi}_2^T(\alpha_2\mathbf{C} + \mathbf{P}_0) = 0$, we have

$$\begin{aligned} & \mathbf{\Pi}_1^T\{[\alpha_1\beta + \alpha_2(1-\beta)]\mathbf{C} + \mathbf{P}_0\} = (1-\beta)\mathbf{\Pi}_1^T[\alpha_2\mathbf{C} + \mathbf{P}_0] \\ = \;& (1-\beta)\frac{\mathbf{\Pi}_1^T\mathbf{P}_0 \cdot \mathbf{\Pi}_2^T\mathbf{C} - \mathbf{\Pi}_2^T\mathbf{P}_0 \cdot \mathbf{\Pi}_1^T\mathbf{C}}{\mathbf{\Pi}_2^T\mathbf{C}} \end{aligned}$$

$$\begin{aligned} \text{and} \qquad & \mathbf{\Pi}_2^T\{[\alpha_1\beta + \alpha_2(1-\beta)]\mathbf{C} + \mathbf{P}_0\} = \beta\mathbf{\Pi}_2^T[\alpha_1\mathbf{C} + \mathbf{P}_0] \\ = \;& \beta\frac{\mathbf{\Pi}_2^T\mathbf{P}_0 \cdot \mathbf{\Pi}_1^T\mathbf{C} - \mathbf{\Pi}_1^T\mathbf{P}_0 \cdot \mathbf{\Pi}_2^T\mathbf{C}}{\mathbf{\Pi}_1^T\mathbf{C}} \end{aligned}$$

Therefore we have

$$(\eta\mathbf{\Pi}_1^T + \zeta\mathbf{\Pi}_2^T)\{[\alpha_1\beta + \alpha_2(1-\beta)]\mathbf{C} + \mathbf{P}_0\} = 0$$

where $\eta = \frac{\mathbf{\Pi}_2^T\mathbf{C}}{1-\beta}$ and $\zeta = \frac{\mathbf{\Pi}_1^T\mathbf{C}}{\beta}$. Apparently, $(\eta\mathbf{\Pi}_1^T + \zeta\mathbf{\Pi}_2^T)^T$ is a plane parallel to $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ and the plane is independent of the choice of $\mathbf{p}$.

We now define a new coordinate system $xy\beta$ with a spatial point's position given by the location of a ray from the reference camera center through pixel at (x, y) and parameter β the position of the point on the line segment of the ray bounded by the two parallel planes.

If we define the distance of a point to one of the bounding planes as the "height" of the point, in the $xy\beta$ system all points with same β have the same height.

If the world frame is the same as the reference camera frame and the bounding planes are orthogonal to the Z axis of the world coordinate system, then each 3-D point with the same β has the same depth.

The β coordinate can be viewed as the height (or depth) in the $xy\beta$ system. In our discretization scheme, we discretize the three coordinates x, y and β. In the space partition scheme, each voxel is indexed by a tuple (i, j, k), which denotes the voxel with center at 3-D point $\mathbf{P}_{i,j,k} = [\alpha_1(\mathbf{p})\beta + \alpha_2(\mathbf{p})(1-\beta)]\mathbf{C} + \mathbf{M}^+\mathbf{p}$, with $\mathbf{p}$ indexed by (i, j) and $\beta = k/K$ where $k = 1..K$. This space partitioning has resolution corresponding to that of the reference image.

3-D Non-Regular Mesh Reconstruction

We assume that each view can see the same set of surface points on the object surface. Based on this assumption, the 3-D mesh will project to a planar 2D mesh in different views with all meshes having the same topology.

A Labeling Problem In the embedded volume, the 3-D mesh reconstruction problem comes down to deciding whether a voxel labeled by the tuple (i, j, k) in the volume V *is on the object's surface or not*. The reconstruction problem can be formulated as the the following *binary* labeling problem:

$$\min_x \sum_{(i,j,k)\in V} c(i,j,k)\cdot x_{i,j,k} + \lambda \sum_{\{(i,j),(m,n)\}\in\mathcal{N}} \sum_{k,l} x_{i,j,k}\cdot x_{m,n,l}\cdot d((i,j,k),(m,n,l))$$
$$s.t.\ \sum_{k=1}^{K} x_{i,j,k} = 1, \quad x_{i,j,k} = 0 \text{ or } 1$$

$\mathcal{N}$ is the set of neighboring pixels in the reference view; $c(\cdot)$ is the cost of one voxel being assigned label 1; $d(\cdot)$ is the distance metric for two voxels; and λ is a weight constant to control the smoothing term.

Here, the labeling cost function $c(\cdot)$ defines the consistency of colors of 3-D point (i, j, k) in different views. For distance function $d(\cdot)$, we take

$$d((i,j,k),(m,n,l)) = w(d_{i,j,m,n})|k-l|$$

where $d_{i,j,m,n}$ is the Euclidean distance between (i, j) and (m, n). The weighting function $w(\cdot)$ is positive and decreasing — this decouples distant points.

Modified Max-flow Method The above problem can be converted to searching for the minimum cut of a suitable network. Let the edge point set in the reference view be R and the Delaunay graph corresponding to these edge points be G. G is the *base graph*. Each edge in G is bi-directed, with capacity of the edge connecting (i, j) and (m, n) in both directions set to $\lambda w(d_{i,j,m,n})$. We generate an array of graphs, where each graph has the same structure as G and the same edge capacities. We introduce a source node s and a sink node t. We connect s to every node in the first layer, and every node in the last layer to t. We further connect corresponding nodes between layers. In the 3-D network we call the edges in each layer *horizontal* edges and the edges between different layers and the source and sink nodes the *vertical* edges. The horizontal edges are bi-directional and vertical edges are uni-directional. We also define a *vertical edge sequence* as the edges from source node to sink node which pass through the corresponding nodes in each layer. We set the capacity of the vertical edges as

$$C_e(e_{i,j,k}) = c(i, j, k) + \Delta(i, j)$$

where $e_{i,j,k}$ is an edge at layer k and connecting node (i, j) between successive layers. Now we show that by selecting suitable $\Delta(i, j)$ we can guarantee that there is a single cutting edge in each vertical edge sequence for a possible min-cut configuration. In the following analysis, we assume that $c(i, j, k) \geq 0$ and the network is constructed with the above method.

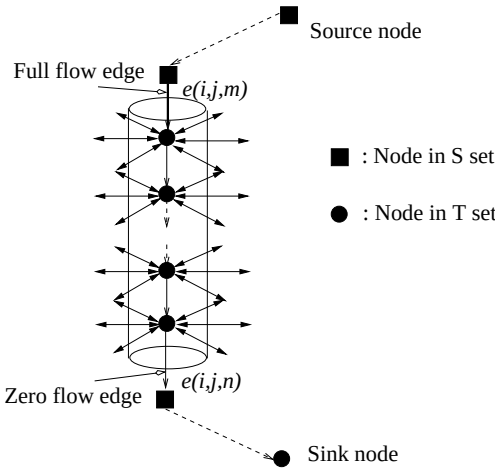


Figure 3.3: A vertical edge sequence of a network.

Observation 3.2. For general planar graph G with outdegree $n(i, j)$ at node (i, j), and

assuming the maximum capacity of out-edges of node $n_{i,j}$ is $H(i,j)$, there is a single cutting edge in each vertical edge sequence if $\Delta(i,j) > n(i,j) \cdot K \cdot H(i,j)$, where K is the number of layers of the network.

Proof: A min-cut of the network partitions the nodes in the network into two sets. The set containing the source node s is defined as the S set and the set containing the sink node t is defined as the T set. We simply need to prove that for a min-cut configuration, if we label all the nodes by their set membership, the nodes in a vertical sequence can only have the pattern $SS...ST...T$ in the ordering from source to sink.

Apparently each vertical sequence must contain a cutting edge, for otherwise s and t will fall into same set. If there is more than one cutting edge in one vertical sequence in some min-cut configuration, there must be a node labeling pattern like $ST...TS$ as a sub-sequence in the labeling of a vertical sequence. Fig. 3.3 shows the segment of nodes with the pattern. Based on the max-flow min-cut theorem, the edge $e(i,j,m)$ must be saturated and the edge $e(i,j,n)$ must have zero flow. Now we consider the net flow of nodes in the cylinder interface. Denoting the incoming flow as F_I and outgoing flow as F_o then we have

$$F_I \geq c(i,j,m) + \Delta(i,j) > n(i,j) \cdot K \cdot H(i,j)$$

$$\text{Since} \qquad F_o \leq n(i,j) \cdot (n-m) \cdot H(i,j) \leq n(i,j) \cdot K \cdot H(i,j)$$

we have $F_I > F_o$ which, means the net flow is not equal to zero and contradicts the net flow condition. Thus there will be one and only one cut in each vertical sequence.

Observation 3.3. If G degenerates to a line graph and the horizontal edges have the same capacities, there is a single cutting edge along each vertical edge sequence if $\Delta(i,j) > 0$.

Proof: Assume some non-boundary vertical edge sequence has more than one cutting edge. A typical configuration is shown in Fig. 3.4. Assuming the boundary starts from the left boundary, it must turn somewhere; apparently, a turn at the right boundary is impossible since if we remove the turn we get an even smaller cut. The "white" set does not include the source or sink node. The edges labeled as "*f*" denote that the flow in these edges is saturated (equal to the capacity of the edge) and "*0*" denotes a zero flow edge. It is easy to see that if the horizontal edges all have the the same capacities, the set of "white" nodes does not have zero net flow, which contradicts the flow condition. Based on a similar argument, we can show that the left boundary vertical sequence can also have only one cut in a min-cut configuration.

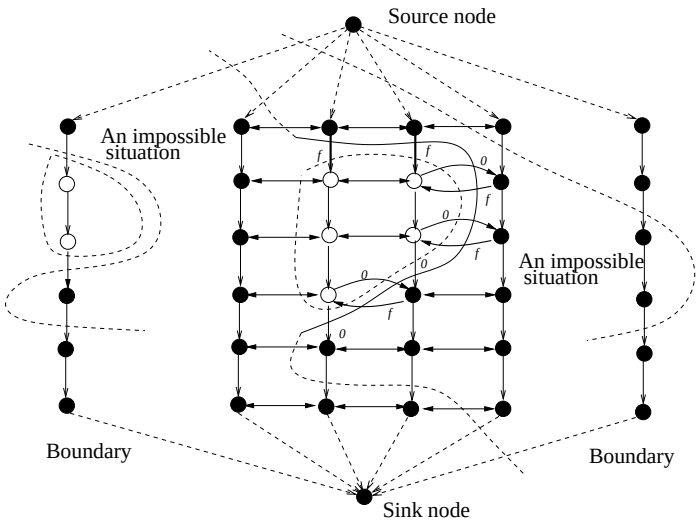


Figure 3.4: Graph cut of a 2D network.

Observation 3.4. If G degenerates to a line graph and the horizontal edges can have different capacities, there will be one and only one cutting edge for each vertical edge sequence if $\Delta(i,j) > K[C_{HMax} - C_{HMin}]$, where C_{HMax} is the maximum horizontal edge capacity and C_{HMin} is the minimum horizontal edge capacity, K is the number of layers of the network. Similarly we have the following:

Observation 3.5. The case when G is a circle graph. There will be one and only one cutting edge for each vertical edge sequence in a min-cut configuration if $\Delta(i,j) > K[C_{HMax} - C_{HMin}]$, where C_{HMax} is the maximum horizontal edge capacity, C_{HMin} is the minimum horizontal edge capacity, and K is the number of layers in the network.

We now show that the min-cut of the network solves the 3-D reconstruction problem. The *cut* of the graph is defined by the sum of the capacities of the forward-direction edges connecting the source node and sink node partitions. We further define the *vertical cut* as the sum of the capacities of the vertical edges connecting the source node and sink node partitions and the *horizontal cut* as the sum of the capacities of the horizontal edges connecting the source node and sink node partitions.

If the vertical edge $e_{i,j,k}$ is a cutting edge then we set $x_{i,j,k} = 1$, else $x_{i,j,k} = 0$; Because for a min-cut configuration there is one and only one single cutting edge along each vertical sequence, the corresponding labeling scheme is feasible, since $\sum_{k=1}^{K} x_{i,j,k} = 1$. The vertical

cut is thus

$$C_v = \sum_{(i,j,k)\in V} [c(i,j,k) + \Delta(i,j)] \cdot x_{i,j,k} = \sum_{(i,j,k)\in V} c(i,j,k) \cdot x_{i,j,k} + \sum_{(i,j)\in R} \Delta(i,j)$$

where R is the edge pixel set of the reference image. Recalling that we assign the capacity of the horizontal edge in G between (m,n) and (i,j) as $\lambda w(d_{i,j,m,n})$, therefore,

$$C_h = \sum_{\{(i,j),(m,n)\}\in\mathcal{N}} \sum_{k,l} x_{i,j,k} x_{m,n,l} \cdot \lambda w(d_{i,j,m,n})|k-l|.$$

Thus the min-cut $C_v + C_h$ has only a constant difference from the energy in the surface reconstruction problem. Finding the min-cut and mapping the resulting labeling variable $x_{i,j,k}$ will can solve the 3-D mesh reconstruction problem.

Surface Refinement

Based on the above max-flow method, we arrive at a reconstructed 3-D mesh which spans the object surface. Usually, the 3-D mesh still needs some further refinement to derive a smooth surface. Firstly, the 3-D surface mesh is usually not "dense" enough. We need to fill holes such that a mesh becomes a surface, using interpolation. Second, the 3-D surface spanned by the mesh usually suffers from staircase effects since we used the Manhattan-like norm in surface reconstruction.

Here, we use the surface spanned by the 3-D mesh as the initial surface, and apply a PDE-based method to refine the surface in a force field determined by the consistency function in different views. The PDE is a simplified version of motion estimation PDE. The PDE is able to adaptive to the surface structure. Similarly, it approaches a standard heat equation with a penalty term of the consistency in different views in smooth surface regions. The diffusion method approaches a mean curvature motion solution at rapidly changing areas such as edges or corners. The height of the surface is now not confined to the levels determined by the non-linear space partition but now can be a real number. A finite difference method is used in the discretization of the PDE.

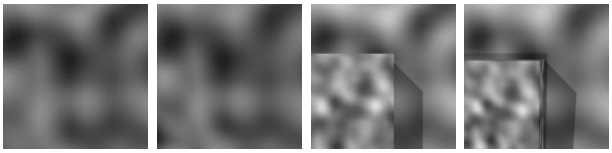
3.4 Experimental Results

In this section, we present the results of the successive convexification motion estimation and structure from motion. In motion estimation, we use only the grayscale image for feature

extraction and matching. If the image is in color, it is first converted to a gray level image before further processing, and normalized to $[0, 1]$.

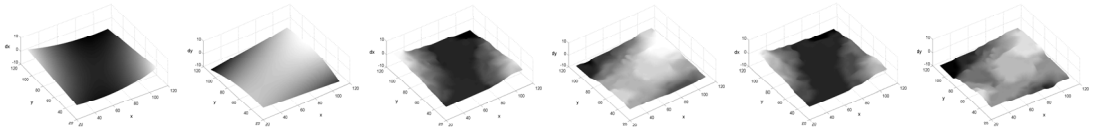
3.4.1 Rotated Plane

We first compare the proposed SC-LP based method with the Graph Cut method for synthesized grayscale imagery with ground truth motion field. The testing images are rendered by 3-D rendering software for two 3-D objects. The first pair of testing images, "rotating plane", are two views of a rotating plane. The second pair of testing images, "overlapping layers", contain abrupt changes of motion fields, which is used to test the detail-preserving property of the methods. The texture in the images is randomly generated and the scale is selected such that there is a great deal of ambiguity in the local matching process. The image size is 128×128. To simplify the experiment, we generate views where there is no occlusion from the reference image to the matching image. The ground truth motion field is calculated in the following manner: First, we calculate the intersection of the ray starting from the reference camera center and passing through a given pixel with the object's surface. We further project the intersection point to the second camera image plane. The motion of the given pixel in the reference image can then be calculated by the coordinate differences in both x and y direction of the original image pixel and the projected image pixel in the matching image. Therefore the ground truth motion is in floating point precision. To simplify the comparison, we ignore occlusion in these experiments for both methods. As well, we use the same cost function and topology for both methods. About 300 and 800 random points are selected respectively in a rectangular area of the reference image for the two experiments and weighting parameters are tuned such that the best performance in the sense of mean absolute error is attained for both schemes respectively. α-expansion is applied for the Graph Cut symbol modification process. The symbol set is all possible motions for each site, over $[-10, 10] \times [-10, 10]$ and $[-12, 12] \times [-12, 12]$ respectively for the two experiments. In this experiment, deterministic symbol change sequence is used in the Graph Cut iteration process and the process iterates until there are no symbol changes. The initial value for Graph Cut is the best local motion searching result. Fig. 3.5 shows the reference and matching images used here. The window size used in the first image pair is 5×5 and the 3×3 for the second image pair. Fig. 3.6 and Fig. 3.7 show the ground truth motion field and the best-performance interpolated motion estimation results for these two experiments, with mean absolute error for the selected points compared in Fig. 3.8. Both



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

Figure 3.5: (a, b): Reference and matching images – `Rotating Plane`; (c, d): reference and matching images – `Overlapping Layers`. The gray parts in the middle of the images are shadows.



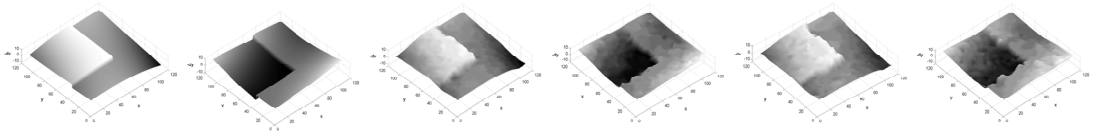
(a) (b) (c) (d) (e) (f)

Figure 3.6: `Rotating Plane`. (a, b): Ground truth x and y motion field; (c, d): interpolated SC-LP motion estimation, smoothing factor = 0.7; (e, f): interpolated GC motion estimation, smoothing factor = 0.9.

the SC-LP and GC method's MAE changes in nearly the same trend with the changing of the smoothing factor: The error decreases until the smoothing factor increases to a certain point and then increases with increasing smoothing factor. For these tests, the SC-LP based method has better performance, both visually and in mean absolute error, as demonstrated in the error curves in Fig. 3.8.

3.4.2 Matching Random Patterns

As another experiment we randomly generate grayscale images and then apply a ground truth deformation model to generate the image for matching. In this experiment, we randomly generated 100 images in each of three different scales and then from each of these we



(a) (b) (c) (d) (e) (f)

Figure 3.7: `Overlapping Layers`. (a, b): Ground truth x and y motion field; (c, d): interpolated SC-LP motion estimation, smoothing factor = 0.4; (e, f): interpolated graph cut motion estimation, smoothing factor = 0.4.

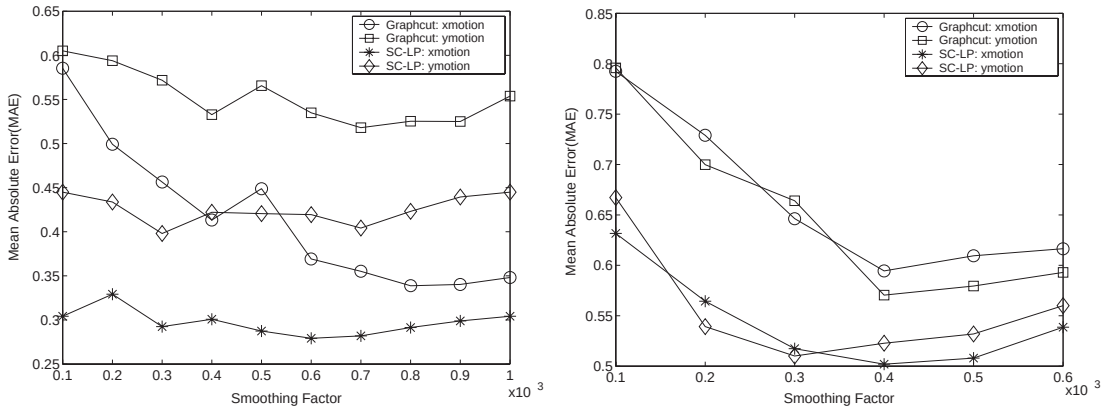


Figure 3.8: MAE Comparison. (a): MAE of dx and dy for Figs. 3.5 (a) and (b); (b): MAE of dx and dy for Figs. 3.5 (c) and (d).

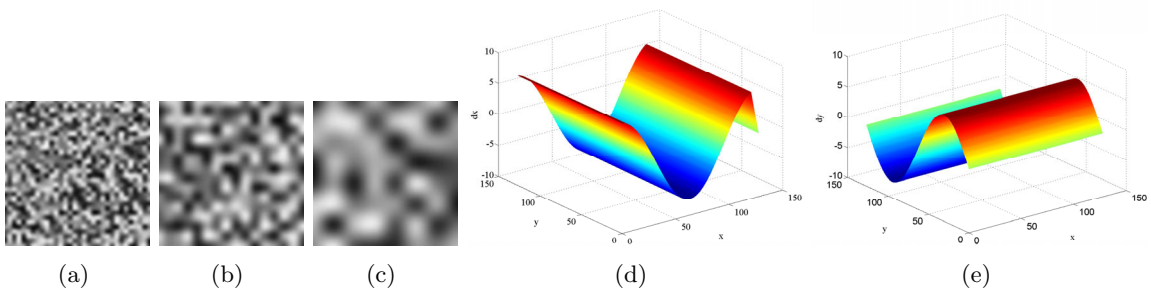


Figure 3.9: Random graylevel images. (a, b, c): Sample template images in the three scales used; (d, e): Deformation model in x and y directions.

also generated test images with a given deformation model. Fig. 3.9 shows the deformation model used in this experiment and three samples of the randomly generated images, at three different scales. We apply the deformation model to the randomly generated images and generate target images for matching. In the reference image for each image pair, about 300 sites are randomly selected. The block size for calculating matching cost is set to be 3×3. The searching window for each site in the reference image is taken to be [-10,10]×[-10,10]. The definition of energy for all methods compared is the same in these experiments. The smoothing coefficients are specified as unity in the matching process. We compare the matching performance of the proposed successive convexification linear programming

Table 3.1: Comparison results for random pattern matching: Scale 1

	SC-LP	GC	BP	ICM
Mean Absolute Error	0.8324	0.9460	1.0099	2.9268
Standard Deviation	0.0737	0.1618	0.1246	0.3892

Table 3.2: Comparison results for random pattern matching: Scale 2

	SC-LP	GC	BP	ICM
Mean Absolute Error	0.8926	1.2452	1.1286	1.8322
Standard Deviation	0.0543	0.4343	0.1324	0.2098

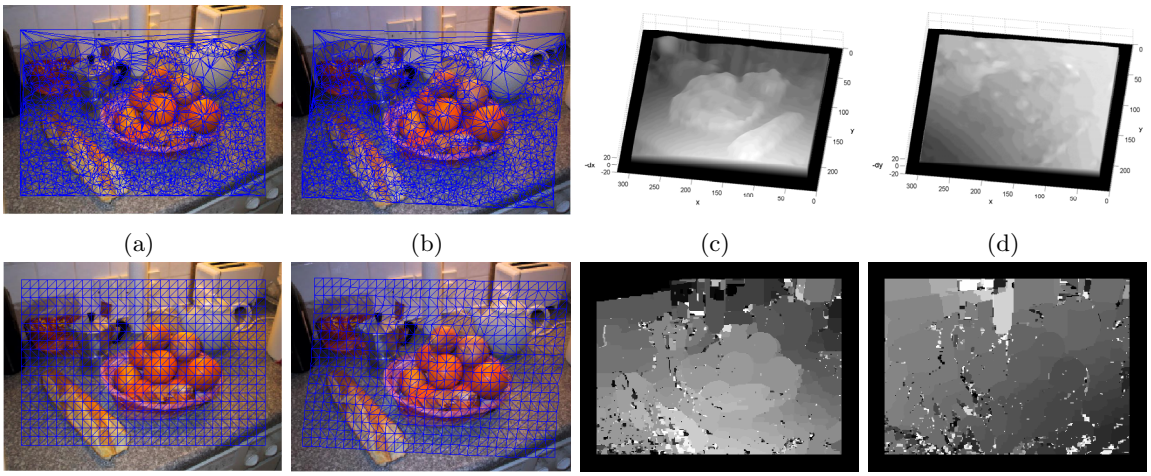
(SC-LP) matching scheme with ICM, the GC and BP for random patterns with 3 different scales. For each of the three scales, 100 random patterns are used in the experiments. Matching error is the mean absolute distance of the matching points to the ground truth target points. Sufficiently large numbers of iterations are set for ICM, the GC and BP such that they converge. We use α expansion for symbol updating in the GC. BP is a baseline implementation without pruning process. The matching error and standard deviation are listed in Tables 1, 2, 3 for the three scales examined. SC-LP achieves the smallest matching error and standard deviation in this experiment. BP and GC have comparable results. ICM is the worst, in this experiment.

3.4.3 Motion Estimation

Fig. 3.10 illustrates the linear programming edge-feature points based matching result and dense motion estimation based on the PDE scheme for image `Table`. A rectangular region has been manually selected as the region of interest. About 2,000 edge-points and supporting points are detected in the region of interest. The local searching region is $[-20, 20]$ in both x and y directions. The cost function is the normalized absolution difference of image blocks. Starting with reference and matching non-regular meshes in Figs. 3.10 (a, b) we calculate the dense motion field in Figs. 3.10 (c, d) based on the PDE scheme. To visualize the

Table 3.3: Comparison results for random pattern matching: Scale 3

	SC-LP	GC	BP	ICM
Mean Absolute Error	1.4954	1.8092	1.7372	2.3045
Standard Deviation	0.2308	0.3230	0.3132	0.3943



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

(e) (f) (g) (h)

Figure 3.10: `Table`. (a): Reference image and mesh; (b): Matching image and matching mesh based on SC-LP; (c, d): Dense x-motion and y-motion based on the proposed two-phase method; (e, f): Regular mesh matching based on dense motion field; (g, h): For comparison, x and y motion estimation based on the GC [36]. Image dataset [89] ©Philip Torr, 2006, by permission.

dense motion estimation result, we plot the matching result on *regular* grids based on the dense motion estimation in Figs. 3.10 (e, f). For comparison, Figs. 3.10 (g, h) show the dense motion estimation result based on the Graph Cut scheme in [36]. More experimental results, using image `Toy_house`, are shown in Fig. 3.11.

In the above experiments, there is little occlusion involved. Fig. 3.12 shows the motion estimation and occlusion inference result for the image pair `Map`. For this image pair, there is only horizontal motion. Fig. 3.13 shows an experiment results for estimating 2-D motion and the occlusion map simultaneously with the proposed scheme. For image `Mouse`, camera motion and object motion are both involved. Another challenge in this experiment is that the mouse and mouse pad contain large areas without texture. The scaled motion vector plot is shown in Fig. 3.13 (c). A threshold of 0.5 is used to obtain the occlusion map in these experiments. In Fig. 3.13 (d), the occlusion map is shown aligned with the reference image by replacing the red channel of the reference image with the occlusion map while keeping intact the green and blue channels. Figs. 3.13 (e, f, g, h) show the matching result based on the proposed linear programming scheme and the dense motion field based on the proposed two-phase motion estimation scheme. Another result for motion and occlusion estimation

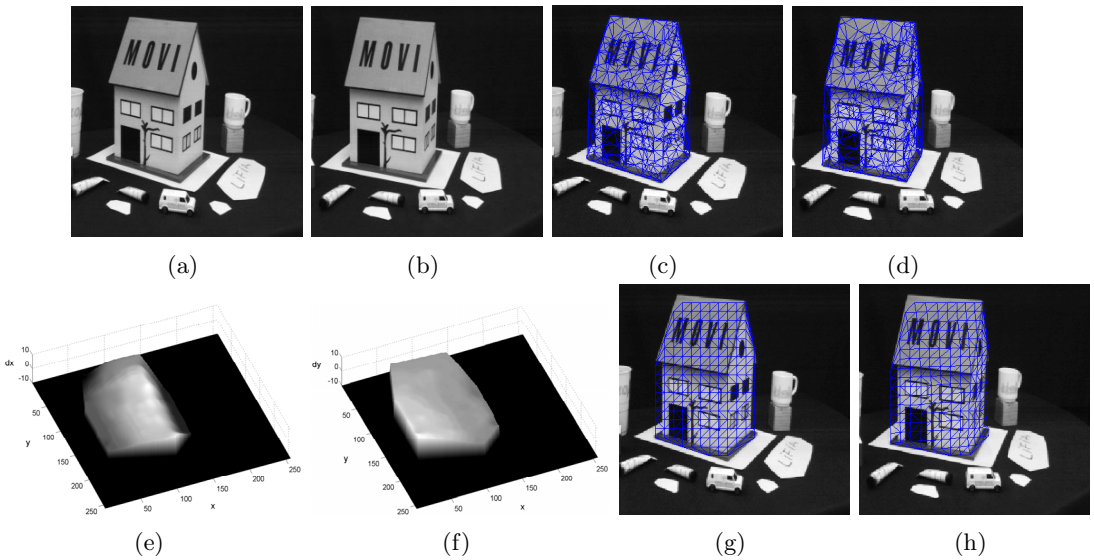


Figure 3.11: `Toy_house`. (a, b): Reference and target images; (c, d): Mesh matching based on the SC-LP method; (e, f): Dense x and y motion; (g, h): Regular mesh matching based on dense motion field.

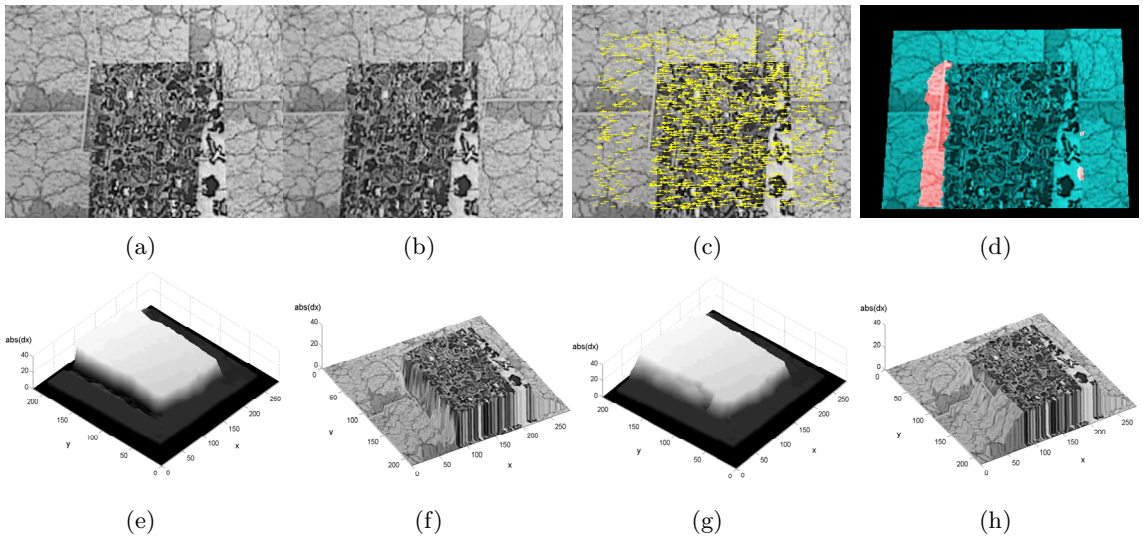
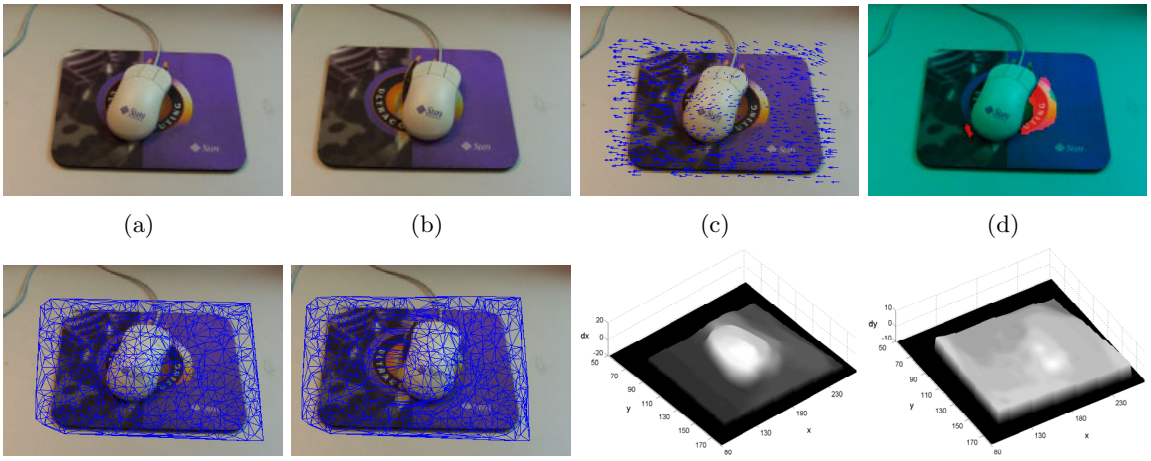


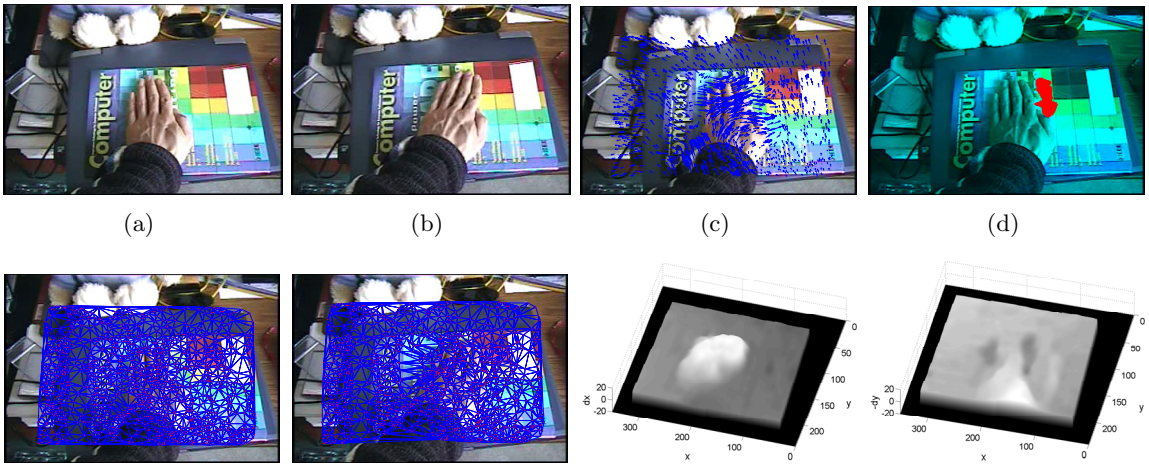
Figure 3.12: `Map`. (a, b): Left and right view; (c): Scaled motion vectors; (d): Occlusion map shown in the red channel; (e, f): Dense disparity map, and texture-mapped figure; (g, h): Dense disparity map *without* occlusion inference, and texture-mapped figure. Image dataset www.middlebury.edu/stereo ©Middlebury College, 2006, by permission.



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

(e) (f) (g) (h)

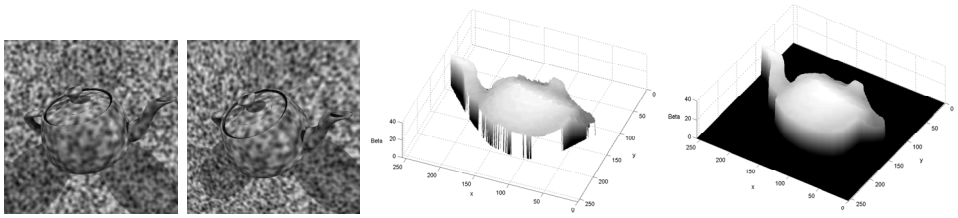
Figure 3.13: `Mouse`. (a, b): Reference and matching images; (c): Scaled motion vectors; (d): Occlusion map shown in the red channel; (e, f): Matching based on SC-LP; (g, h): Dense x-motion and y-motion based on the proposed two-phase method.



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

(e) (f) (g) (h)

Figure 3.14: `Hand`. (a, b): Reference and matching images; (c): Scaled motion vectors; (d): Occlusion map shown in red; (e, f): Matching based on SC-LP method; (g, h): Dense x-motion and y-motion based on the proposed two-phase method.



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

Figure 3.15: (a, b): Two views of a synthesized scene; (c): Interpolated surface by the maximum-flow method; (d): Refined surface by the proposed PDE method.

is shown in Fig. 3.14.

3.4.4 Structure from Motion

Experimental results are based on both synthesized and natural images. Fig. 3.15 shows two views of a synthesized scene, with random texture applied to surfaces. We set the left image be the reference image and the world coordinate system be the reference camera's frame, with two bounding planes parallel to the imaging plane. In this experiment, the camera matrix is known for each view and we would like to reconstruct the 3-D surface based on the hierarchical scheme. We took 40 samples along the β axis, and x and y axis have sampling equal to the pixel resolution. In this experiment the ROI (region of interest) is the teapot and the mask is obtained manually from the reference image. In the first step, edge detection produces edge pixels consisting of about 1/5 of the pixels in the ROI. In the second step of the reconstruction, we use the stated sampling resolution along the β axis. To deal with the boundaries of the ROI during surface evolution, we apply padding by extending the boundary pixels horizontally and vertically to the region outside the object. Fig. 3.15 shows the surface spanned by the reconstructed 3-D mesh and the surface refinement result based on our surface evolution method. The surface is represented in $xy\beta$ space.

Fig. 3.16 shows views of a natural scene, with the object of interest placed on a calibration checkerboard. In this experiment, we captured a total of 6 images of the object, and calibrated the camera intrinsic and extrinsic parameters based on these images. However, we use only two of them in the reconstruction process. Here, the world coordinate system x, y plane is fixed on the calibration checkerboard and bounding planes are parallel to it. The ROI in the reference image is a rectangular region on the slanted newspaper surface. Fig. 3.16 shows the surface reconstructed from the first and second steps of the proposed method,

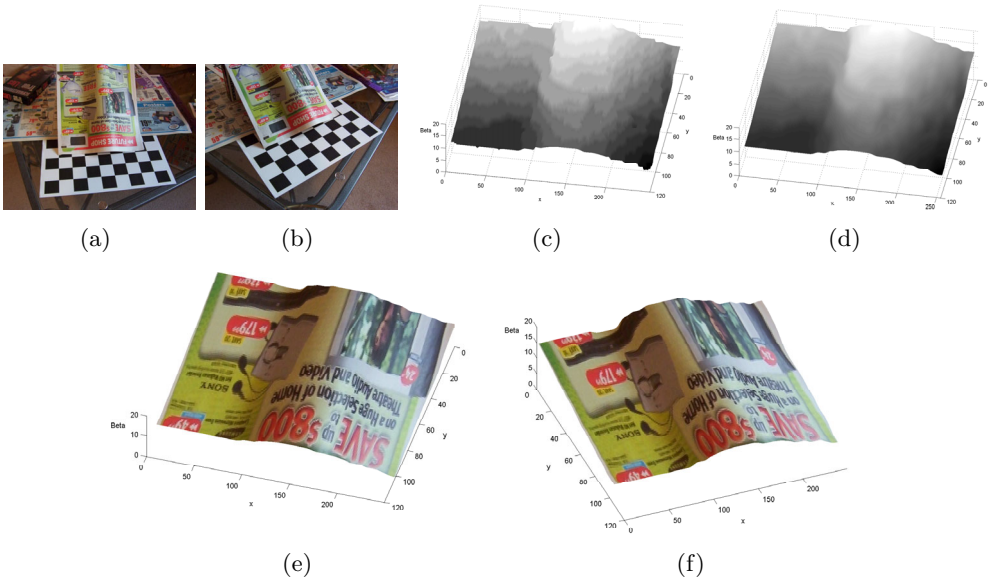
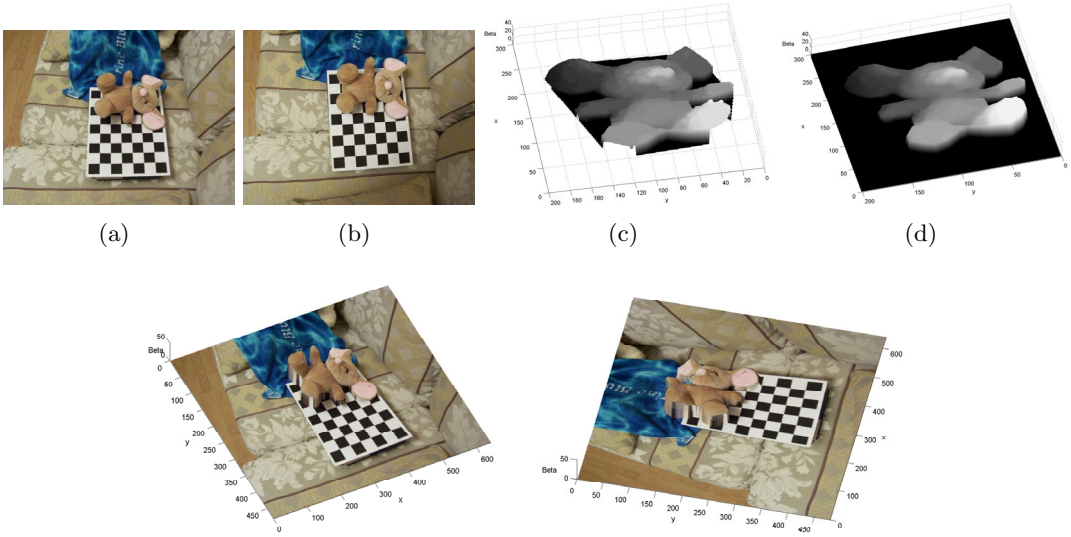


Figure 3.16: (a, b): Two views of a natural scene. (c): Interpolated surface by maximum-flow method; (d): Refined surface from PDE smoothing; (e, f): Texture mapped surface.

and Fig. 3.16 shows the texture-mapped ROI in two new views. For another experiment, Fig. 3.17 shows the first two images and the corresponding reconstructed, texture-mapped result. In Fig. 3.17, only the mouse (ROI) has height information.

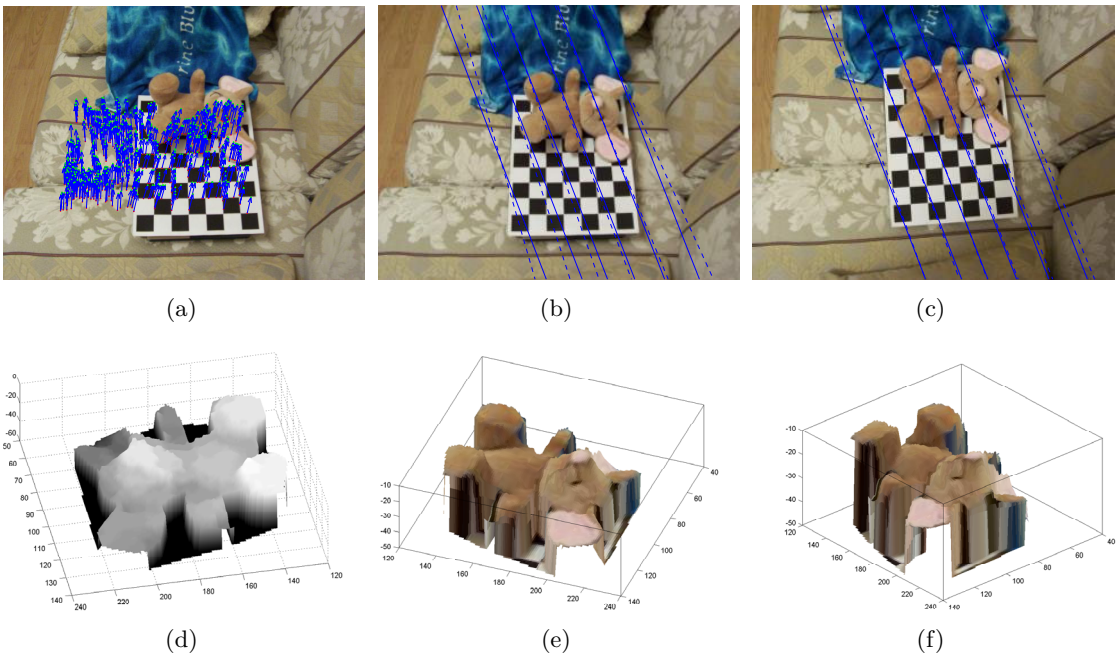
We further applied the method to shape reconstruction using uncalibrated images. The successive convexification scheme is used to calculate the motion vectors between two views. Fig. 3.18 (a) shows the matching feature points for the images of `Stuffie` — red feature points in the reference image are connected to corresponding green points in another image. In Figs. 3.18 (b, c), the epipolar lines based on the estimated fundamental matrix from the matching points are shown as solid lines and the dashed lines show the epipolar lines estimated from camera calibration using the checkerboard, based on Tsai's method [82]. We assume a zero-skew intrinsic parameter matrix in this experiment. The reconstructed 3-D shape based on the estimated camera external parameters is shown in Fig. 3.18 (d). The 3-D surface reconstruction is based on 2-step hierarchical scheme. Figs. 3.18 (e, f) show two different views generated from the texture-mapped 3-D reconstruction result. Another camera calibration and 3-D structure reconstruction experiment result is shown in Fig. 3.19.



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

(e) (f)

Figure 3.17: (a, b): Two views of another natural scene; (c): Interpolated surface by maximum-flow method; (d): Refined surface from PDE smoothing; (e, f): Texture mapped surface.



(a) (b) (c)

(d) (e) (f)

Figure 3.18: `Stuffie`. (a): Matching points; (b, c): Epipolar lines in reference and matching image; (d): Shape reconstructed based on the sparse max-flow and PDE methods; (e, f): Two different views, texture-mapped.

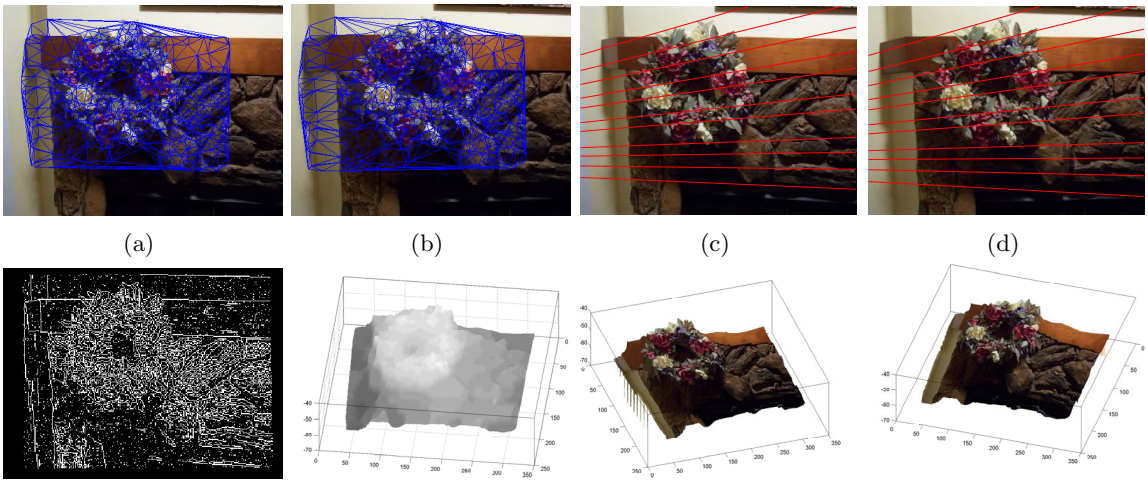


Figure 3.19: `Flower`. (a, b): Image matching result based on LP; (c, d): Epipolar lines in the reference and matching images; (e): The feature points used in max-flow mesh reconstruction; (f): Shape reconstructed based on the max-flow and PDE methods; (g, h): Two different views of texture-mapped result.

3.5 Summary

In this chapter, we study an extension of the successive convexification scheme with occlusion/outlier inference. The convex relaxation shares many similar properties with the basic consistent labeling formulation. We can still use a small number of basis labels in the linear programming construction. For each site, these basis labels correspond to the vertices of the lower convex hull for the motion cost surface. The successive convexification schemes can also be applied. The proposed method yields motion estimation and occlusion map simultaneously. Experiments show that the successive convex motion estimation yields smaller errors than other matching schemes such as ICM, BP and Graph Cut for motion estimation using ground truth data. Together with a detail preserving PDE, we are able to estimate dense motion for large scale motion estimation problems. In the two-step method, a pseudo-dense motion estimation result is first estimated using the successive convex motion estimation; a detail preserving PDE is then used to upgrade the result into dense motion fields. Detail preserving PDE reconstruct motion map isotropically at smooth regions, and mostly along the edges for regions with details. The PDE is adaptive to the local structures of the initial estimation that is obtained by successive convexification. Therefore it is able

to reconstruct the motion map with sharp discontinuities. We further study reconstructing 3-D shapes based on a hierarchical scheme. A maximum flow scheme is devised for mesh reconstruction. Combining with successive convexification motion estimation and a detail preserving PDE, we can reconstruct dense surface shapes from two uncalibrated images.

Chapter 4

Successive Convexification for Object Tracking

Tracking is an important task for vision applications. In this chapter we study robust object tracking methods. Before we study the tracking scheme based on successive convexification, we first study an inertia snake model for contour tracking, especially for tracking objects in strong shadow interference. This PDE based method is light weight because it only uses the boundary information. It is more robust than traditional active contour schemes, but still suffers from robustness problem when tracking objects in strong clutter. This is due to PDE's local optimization property. Successive convexification method can be used to solve the problem much more robustly. We propose an appearance adaptive tracking method based on the successive convexification scheme. We also propose a method to combine the appearance based mesh tracking with contour tracking, which can also be solved with successive convex relaxation. As illustrated in Fig. 4.1, this chapter extends successive convexification to sequential labeling problems and studies a robust tracking scheme.

4.1 The Inertia Constraint Snake

In this section, we study an inertia constraint snake for contour tracking [43]. Two new external forces are introduced into the snake equation based on the *predictive contour constraint* and the *chordal length constraint* such that the active contour is attracted to a shape similar to the one in the previous video frame. The discrete representation is quite lengthy.

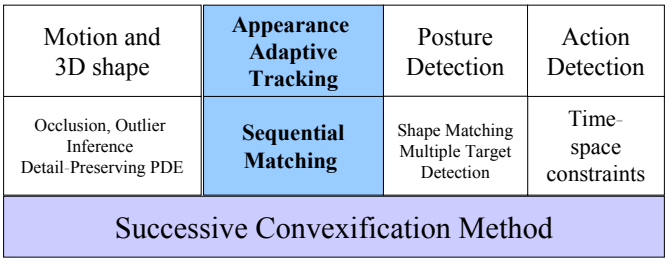


Figure 4.1: Successive convexification for object tracking.

We use a more compact continuous formulation here.

Assuming that the contour is represented as $X(s) = [x(s), y(s)], s \in [0, 1]$, contour tracking is the process of estimating $X(s)$ at current instant based on its contour $X^{-1}(s)$ in the previous frame.

We would like to estimate $X(s)$ by optimizing the following variational problem:

$$\min_X \int_0^1 \frac{\alpha}{2}\|\nabla X(s)\|^2 + \frac{\beta}{2}\|\nabla^2 X(s)\|^2 + P(X(s)) + \frac{\gamma}{2}e(X(s), C(s)) \\ + \frac{\rho}{2}e(\|X(s) - X(s+1/2)\|, d_{X^{-1}}(s))ds$$

where $P(\cdot)$ is a function that has smaller values at motion boundaries; $C(s)$ is a predictive contour; Diameter in the previous frame $d_{X^{-1}}(s) = \left\|X^{-1}(s) - X^{-1}(s+1/2)\right\|$; α, β, γ and ρ are regularization coefficients; function $e(\cdot, \cdot)$ is defined as

$$e(A(s), B(s)) \;=\; \|A(s) - B(s)\|^2.$$

In a Euclidean norm, the resulting Euler equation is

$$-\alpha X_{ss} + \beta X_{ssss} + \nabla P(X) - \gamma(C - X) - \rho\frac{(X(s+1/2) - X(s))}{\|X(s) - X(s+1/2)\|} \\ (\|X(s) - X(s+1/2)\| - d_{X^{-1}}(s)) \;=\; 0$$

Using a fictitious time variable t, the resulting iterative steepest descent solution is the following PDE:

$$\frac{\partial X}{\partial t} \;=\; \alpha X_{ss} - \beta X_{ssss} + F_{ext}(X) + \gamma(C - X) \\ + \rho\frac{(X(s+1/2) - X(s))(\|X(s) - X(s+1/2)\| - d_{X^{-1}}(s))}{\|X(s+1/2) - X(s)\| + \varepsilon}$$

where ε is a small positive number to avoid zero denominator; $-\nabla P(X)$ in the above equation is replaced by a generalized force term $F_{ext}(X)$ that is determined by image features,

Figure 4.2: Tracking result with the inertia snake for the `baby` sequence, selected from a 60-frame sequence.

e.g. motion boundaries. Here we augment $X(s)$ with an artificial time t and the PDE corresponds to iterative greedy search starting from a suitable initial contour. On the right hand side of the PDE, the first two terms represent internal constraints and the last two terms represent predictive contour constraint and chordal length constraint respectively. We use finite difference method to discretize the PDE into difference equations.

We have applied this greedy method to shadow resistant object tracking [43]. External force F_{ext} is based on motion boundaries which are estimated using original video sequence and illumination normalized video sequence. We use an affine global motion model. The predictive contour is obtained using ICM. For short image sequences, this light weight tracking scheme works fine. One tracking result for a baby face is shown in Fig. 4.2. Figs. 4.3 and 4.4 show outdoor traffic scene tracking results.

This simple tracking scheme only uses the boundary information. For complex videos where there are cluttered backgrounds or multiple move objects, such simple schemes do not work well. The incremental scheme also suffers from drifting problem: errors accumulate and fail the tracking. A natural extension is to track meshes instead of contours so that we can take advantage of texture information inside the object boundaries. Mesh tracking also generates accurate point-wise correspondence. In the following, we study mesh tracking based on the proposed successive convexification scheme.

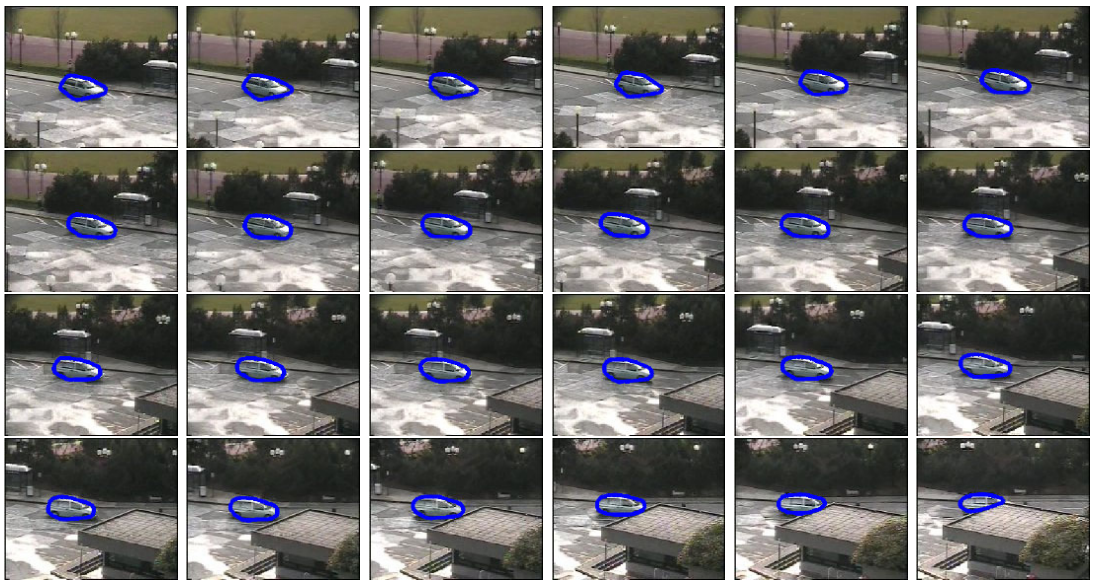


Figure 4.3: Tracking result with the inertia snake for the `car` sequence, selected from a 100-frame sequence.

Figure 4.4: Tracking result with the inertia snake for the `traffic` sequence, selected from a 100-frame sequence.

4.2 Successive Convexification for Mesh Tracking

In this section, we study a more robust tracking method based on the successive convexification scheme. We formulate tracking as a mesh matching problem. Instead of simply using the information at object boundaries, we track objects based on their appearance, i.e. color, texture and their spatial relations. This tracking scheme is more robust than the pure contour tracking method and can track object robustly in thousands of frames.

Mesh Tracking can be treated as a special labeling or matching problem in which we would like to sequentially find the point-wise correspondence for a bunch of template and target image pairs. Because of its sequential property, we can design more efficient methods for this matching (labeling) problem.

4.2.1 Object Representation

In mesh tracking, an object is represented as a mesh (a graph), in which nodes are image features and edges denote neighbor relations. Here, the features are image blocks in the template images and target images. Mesh tracking is thus the problem of finding correspondence of mesh nodes in a sequence of video frames. We consider online tracking in which we can only use information of current video frame and all the tracking histories of previous frames.

Here, we would like the search range for each node on the mesh to be the whole target video frame. Since successive convexification is used in matching, the computational load of optimization is in fact largely decoupled with the sampling density of target feature points. Theoretically, we can use all the image pixels in a target image as candidate points. To reduce the computation load in calculating matching costs, we restrict features all centered on edges both in template image and target video frames. We use a very low edge detection threshold so as not to lose weak features.

To make the scheme resistant to illumination changes, we use chromaticity color space, in which the three color channels are normalized by their arithmetic mean,

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}.$$

And, L_1 norm is used in calculating the cost of matching image blocks.

Image blocks at boundaries need special treatment. Since the boundary blocks may partially contain background pixels, these features are not reliable for object tracking. We

use non-square blocks at the boundary of the template. In fact, we only keep the mask of the whole template object and we can identify the interior points when calculating the matching costs. The average pixel color difference within the mask of small image blocks is then calculated and used as the cost for a potential matching. We use Delaunay triangulation to obtain the neighbor relations of the feature nodes.

4.2.2 Tracking Meshes

Based on such an object representation, object tracking can be done by sequentially matching object in current frame based on previous frame's tracking result. We then update the template based on the matching and proceed to the next video frame. Matching based on successive convexification can be directly applied to such a tracking scheme. We do not have to deal with rotation and scale changes, since these changes are usually small in successive video frames. By updating the template from frame to frame, sequential matching can be done and used to generate the tracking result. Unfortunately, this simple incremental tracking method does not work robustly in real situations. Matching error accumulates in each step and fails the tracking gradually.

A different approach for mesh tracking is as follows. We choose exemplars of object's key appearances and generate key frame templates. Mesh tracking can be done by setting up node correspondence between template object and the target in each video frame. Here, we have to deal with scale and rotation of objects. We also need to deal with large appearance changes. We study this problem in the section of appearance adaptive tracking.

Similar to object localization in Chapter 2, we decompose the geometrical transformation of a template into two cascaded transformations: a global transformation $\mathcal{G}^{(n)}$ (rotation and scaling) and a local deformation $\mathcal{D}^{(n)}$, in which n indicates the frame number. The global transformation is shared by all the sites in the template. In such a model, the matching cost function $c(\cdot)$ is parameterized by global transformation and we rewrite $c(\cdot)$ as $c_{\mathcal{G}^{(n)}}(\cdot)$. $c_{\mathcal{G}^{(n)}}(\cdot)$ is a function of source pixel (site) and target pixel (label). The object tracking problem in frame n becomes an energy minimization problem:

$$\min_{\mathcal{G}^{(n)},\mathcal{D}^{(n)}} \left\{ \sum_{\mathbf{s}\in S} c_{\mathcal{G}^{(n)}}(\mathbf{s}, \mathcal{D}^{(n)} \circ \mathcal{G}^{(n)}(\mathbf{s})) + \lambda \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}} ||\mathcal{D}^{(n)} \circ \mathcal{G}^{(n)}(\mathbf{p}-\mathbf{q}) - \mathcal{G}^{(n)}(\mathbf{p}-\mathbf{q})|| \right\}$$

In tracking, we need to update the global transformation $\mathcal{G}^{(n)}$, the rotation angle and scaling factor, after each matching process. With $\mathcal{G}^{(n)}$ fixed, the optimization is reduced to a

standard consistent labeling problem and we can solve $\mathcal{D}^{(n)}$ by the successive convexification method. Different from object location problem, we do not need a complex exhaustive searching process to determine $\mathcal{G}^{(n)}$. Assuming the initial rotation and scale to be known, we can use the following sequential update procedure.

The global transformation updating process includes two steps: prediction and modification. Predication is based on values in previous two instants:

$$\begin{aligned}
\theta_{pred}(n) &= 2\theta(n-1) - \theta(n-2) \\
\gamma_{pred}(n) &= 2\gamma(n-1) - \gamma(n-2) \\
\theta(-1) &= \theta(-2) = 0 \\
\gamma(-1) &= \gamma(-2) = 1
\end{aligned}$$

The template's global deformation is then set based on predictive rotation θ_{pred} and scale γ_{pred}. We use the successive convex matching to find point to point correspondence from the template to the target video frame. Based on the matching, we re-estimate the rotation angle θ_{est} and scaling factor γ_{est}. The rotation θ and scale γ are modified based on the following smoothing model, in which α is typically 0.9:

$$\begin{aligned}
\theta(n) &= \alpha\theta_{est}(n) + (1-\alpha)\theta_{pred}(n) \\
\gamma(n) &= \alpha\gamma_{est}(n) + (1-\alpha)\gamma_{pred}(n)
\end{aligned}$$

To estimate $\theta_{est}(n)$ and $\gamma_{est}(n)$, we can solve a linear fitting problem. For each corresponding point pair (x_i, y_i) and (u_i, v_i), $i = 1..K$, on the template and target object respectively, we have

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} x_i & y_i & 1 & 0 \\ y_i & -x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} sc \\ ss \\ tx \\ ty \end{bmatrix}, i = 1..K$$

where

$$sc = \gamma_{est}(n)\cos(\theta_{est}(n)), \quad ss = \gamma_{est}(n)\sin(\theta_{est}(n)),$$

We can solve $[sc, ss, tx, ty]^T$ based on the over-constraint linear system. It is then straightforward to calculate $\gamma_{est}(n)$ and $\theta_{est}(n)$ from sc and ss.

The proposed tracking scheme is resistant to drifting because we do not track object sequentially based on features in previous frames, and thus avoid template errors accumulated during the tracking process. But we do use parameters such as rotation angle and scale estimated from previous frames to reduce the searching complexity. A model selection error may still possibly spread to future frames and make the tracker fail. In our scheme, tracking failure can be detected by comparing the minimum matching error with a threshold. When the matching error is too large, we infer a tracking failure and apply a restart process.

Algorithm 4.2: *Successive Convexification for Mesh Tracking*

1. Construct the graph template;
   $\theta(-1) = \theta(-2) = 0$; $\gamma(-1) = \gamma(-2) = 1$; $n = 0$;
2. While $(n < N)$
3. Compute $\theta_{pred}(n)$ and $\gamma_{pred}(n)$ and update the template;
4. Template to object correspondence based on successive convexification;
5. Re-estimate $\theta_{est}(n)$ and $\gamma_{est}(n)$;
6. Update estimation of $\theta(n)$ and $\gamma(n)$ based on the prediction and re-estimation result;
7. $n \leftarrow n + 1$;

Fig. 4.5 shows an experiment result for tracking a planar object under indoor lighting conditions. The glossy surface makes robust tracking a challenging task. In this experiment we use the first frame as the template image. The region of interest is selected manually and a graph template is generated automatically based on random selected edge pixels and Delaunay triangulation. For each video frame, search range is the whole target image. All the edge pixels in a target video frame are potential matching candidates. The scale and rotation of the template are adaptively updated based on the matching result with successive convexification. In the experiment, the smoothing factor α for scale and rotation updating is set to 0.9. The proposed scheme successfully tracks the object over the 400-frame video sequence. The successive convexification scheme is crucial for the success of tracking. Since object's appearance could change quite much from the key frame template, object tracking based on simple greedy schemes does not work well. As shown in Fig. 4.6, the tracker loses the object in a few hundred frames because of the matching failure of the greedy scheme. Other robust matching schemes such as BP is too slow for this application because of the large number of target features points (about 8,000). The successive convexification

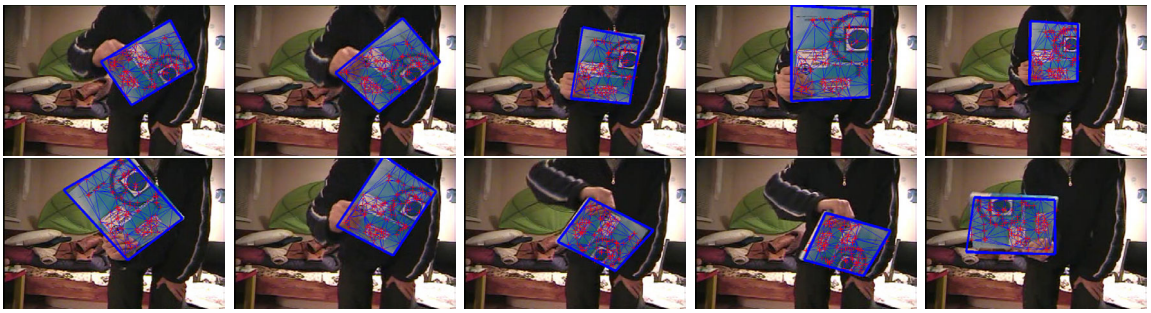


Figure 4.5: Tracking result for the `book` sequence with the proposed successive convexification scheme. Selected frames from 400 frames.

Figure 4.6: Tracking result with ICM for the `book` sequence.

scheme's complexity is comparable to ICM but is much more robust.

A face tracking result is shown in Fig. 4.7. The proposed tracking scheme accurately finds the correspondence of mesh nodes in the video sequence. In Fig. 4.8, a video tape is tracked in a 600-frame sequence. The object in the video undergoes large rotation and scale changes. Fig. 4.9 shows another tracking result based on a single template, this time for a cluttered outdoor scene. The back of the car has little texture which makes tracking a hard problem. The object undergoes rapid motion because of the shaking of camera. The lighting also changes dramatically when the car passes a bus. The proposed successive convex programming based scheme robustly and accurately follows the moving car in a video sequence with 1000 frames.

4.2.3 Appearance Adaptive Object Tracking

Based on the above mesh tracking framework, we further proposed a multiple template tracking method that can be used to track objects with large appearance changes. We use a set of templates to represent possible object appearances in the tracking process. Object matching is based on the proposed successive convex matching method. The algorithm is able to automatically select the best key appearance template and locate the object in the video by fitting the best template mesh to the tracking targets. Because of the robustness

Figure 4.7: `Foreman`. Video tracking result based on the proposed successive convexification method. Selected from 100 frames.

Figure 4.8: Tracking result for video tape sequence. Selected from 600 frames.

Figure 4.9: Tracking result for car sequence. Selected from 1000 frames.

and efficiency of the successive convex matching scheme, object searching can be done in the whole video frame; this enables the method to be applied to tracking problems involving fast motions. The template selection is straightforward after finding the matches of the feature points in the template with corresponding points in the target image based on the proposed method. We use the linear combination of matching cost and deformation measures defined in Chapter 2 to quantify the difference between the template and its matching target.

One assumption we use in the tracking process is that the template's scale and rotation remain continuous even if the template changes. This assumption is valid for appearance changes of most real-world objects, from simple rigid objects to complex articulated ones. To further simplify the algorithm and constrain the searching, these templates can be formulated as a digraph to represent the possible transitions from one appearance model to another. Models that can be reached in one step from the current model include the model itself and its neighbors. Other parameters in the tracking include the scale and rotation changes of the template. Based on this formulation, tracking becomes the process of locating the object with the best templates constrained by the model transition graph.

Algorithm 4.3: *Successive Convexification for Appearance Adaptive Tracking*

1. Construct J Graph templates;

   PrevTemplate $= 1$;

   $\theta(-1) = \theta(-2) = 0$; $\gamma(-1) = \gamma(-2) = 1$; $n = 0$;

2. While $(n < N)$
3. Compute $\theta_{pred}(n)$ and $\gamma_{pred}(n)$ and update the template;
4. for $m = 1$ to J
5. If m is neighbor of PrevTemplate
6. Template to object correspondence based on successive convexification;
7. Compute deformation D and matching degree M;
8. If m has a smaller $M + \alpha D$
9. $Temp = m$;
10. Store matching result;
11. PrevTemplate = Temp;
12. Re-estimate $\theta_{est}(n)$ and $\gamma_{est}(n)$ based on the best matching template's correspondence;
13. Update estimation of $\theta(n)$ and $\gamma(n)$ based on the prediction and re-estimation result;
14. $n \leftarrow n + 1$;

Fig. 4.10 shows a tracking result in which two exemplars are used. The hand undergoes dramatic shapes changes between the two gestures. There are also large scale and rotation changes of the hand involved in this sequence. The proposed scheme successfully tracks the movement of the hand in a poor, low-contrast video. Fig. 4.11 shows a result for tracking a walking person, using three exemplars. The posture of the person walking in the scene is accurately recovered. The template follows the object successfully in this very complex-background setting. In these experiments, the objects in video contain very little texture, which makes salient feature based schemes such as SIFT method not robust in these applications. The proposed scheme utilizes the consistency of matching for many small image patches and is more robust for tracking objects with little texture than salient feature based schemes.

4.3 Combining Contour Tracking and Mesh Tracking

Mesh tracking is usually not very accurate in extracting object's boundaries. Active contour methods locate object's boundary more precisely but suffer from robustness problem when the background clutter increases. Here, we study a contour tracking scheme by integrating

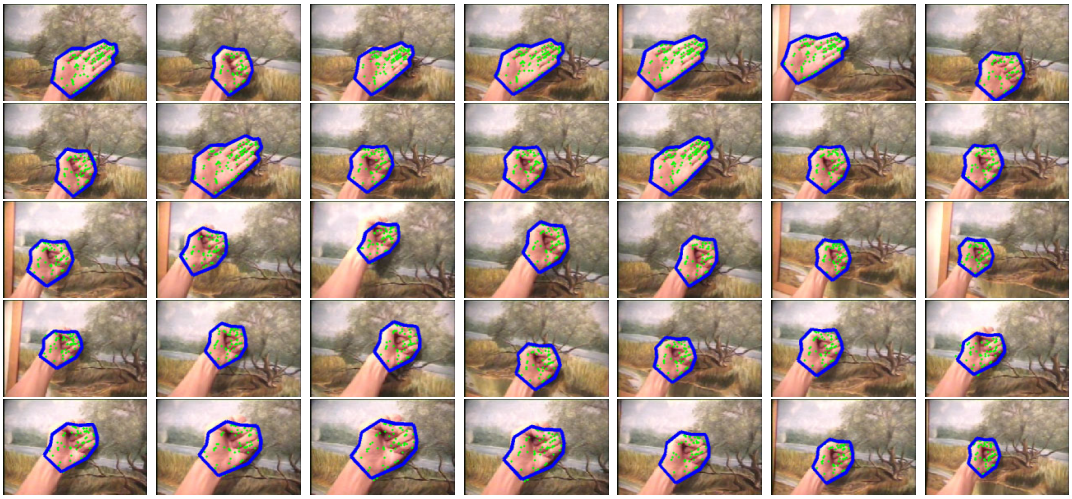


Figure 4.10: Tracking result for hand sequence. Selected from 500 frames.

the information of interior object's texture which has not been fully utilized in traditional contour tracking schemes. As has been studied in the previous section, we can track a cluster of points inside the object boundary robustly with the proposed successive convexification method. We use these points as reference points to improve the robustness of boundary extraction.

In the following, we study a method for refining object's boundary based on the successive convexification method. It is usually not robust to directly match image patches at object boundaries. Such image patches usually contain both object pixels and background pixels. When object moves, the background will change and thus makes these features unreliable in matching. Even though we can use masking techniques studied in the previous sections, the real boundary is hard to locate very accurately in tracking since the colors at object boundary are not stable. Here we consider a different approach. The edges at object boundary are found to be good features in boundary tracking. These boundary edges are easy to detect and usually form a smooth continuous curve. Our goal is to find a continuous object boundary that is consistent with the object movement based on the internal feature points. We assume that we set the initial contour and object internal regions. We assume the template contour is $\mathbf{p}(k), k = 0..K-1$. We would like to estimate the object boundary contour in successive video frames.

Based on the tracking results of the internal feature points, we estimate an affine transformation $\mathcal{A}$ from the matching. $\mathcal{G}$ is the global transformation defined in the previous

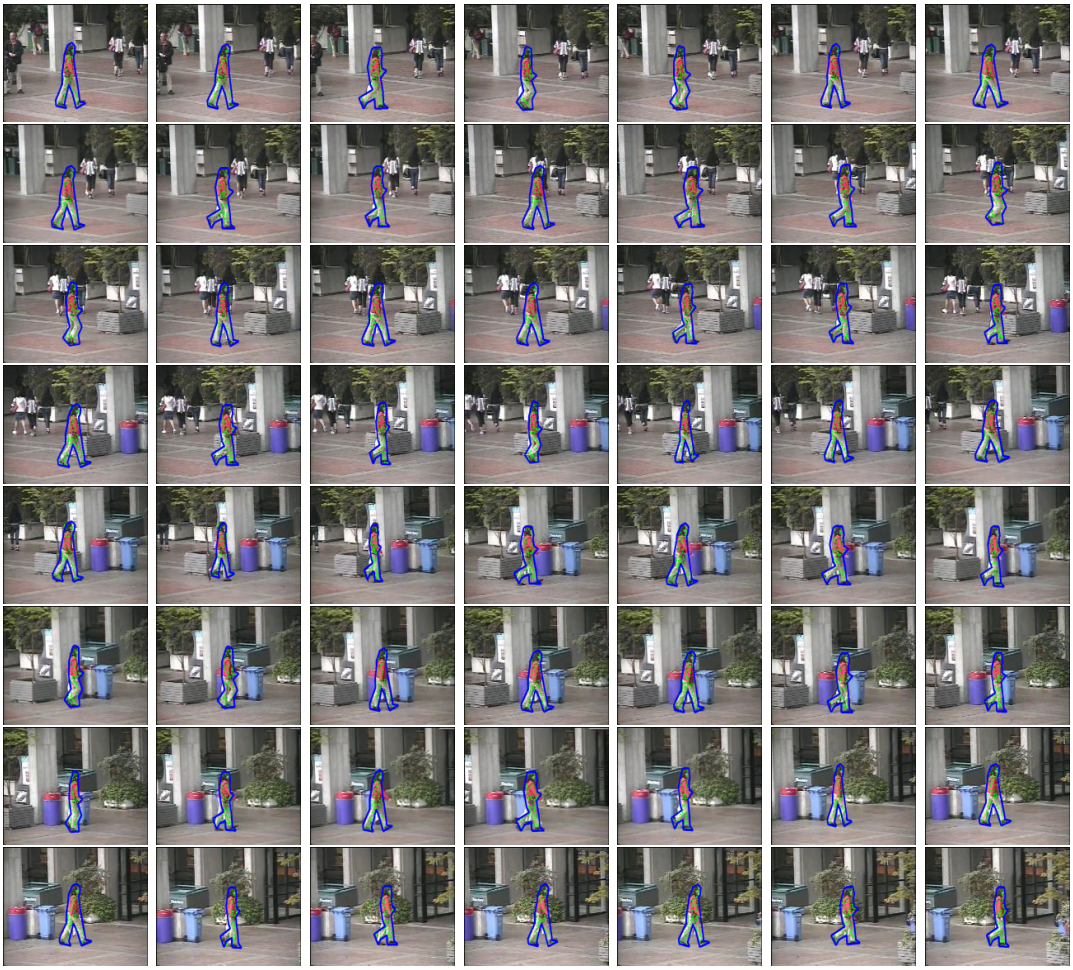


Figure 4.11: Tracking result for walking sequence. Selected from 150 frames.

section. From the internal feature points matching, we try to predict the boundary node position in the target frame by $\mathcal{A}\mathbf{p}(k), k = 0..K-1$. Denoting $\mathbf{f}(k), k = 0..K-1$, to be the unknown boundary contour, we would like to minimize the following energy

$$\min_{\mathbf{f}} \left\{ \sum_{k=0}^{K-1} ||\mathcal{A}\mathbf{p}(k) - \mathbf{f}(k)|| + \lambda \sum_{\{\mathbf{p}(i),\mathbf{p}(j)\} \in Neighbors} ||\mathbf{f}(i) - \mathbf{f}(j) - \mathcal{G}(\mathbf{p}(i) - \mathbf{p}(j))|| \right\}$$
$$s.t. \quad \mathbf{f}(k) \in L, \text{ the edge point set in target frame.}$$

The neighbor relation is defined by the edges of the Delaunay graph of point set $\{\mathbf{p}(k), k = 0..K-1\}$ and therefore one point may have more than 2 neighbors; $||.||$ is the L_1 norm. This is a discrete optimization problem. But we can convert this problem into successive convexification linear programming. It is not hard to reformulate the problem into the following familiar linear program

$$\min \left\{ \sum_{k=0}^{K-1} \sum_{\mathbf{t} \in L} c(k, \mathbf{t})\, \xi_{k,\mathbf{t}} \; + \; \lambda \sum_{\{\mathbf{p}(i),\mathbf{p}(j)\} \in \mathcal{N}} \sum_{m=1}^{2} (f^{+}_{i,j,m} + f^{-}_{i,j,m}) \right\}$$

with constraints:

$$\sum_{\mathbf{t} \in L} \xi_{k,\mathbf{t}} = 1 \ , \ k = 0..K-1$$
$$\sum_{\mathbf{t} \in L} \xi_{k,\mathbf{t}}\, \phi_m(\mathbf{t}) = f_{k,m}, \ , \ \phi_m(\mathbf{t}) = (m\text{th component of } \mathbf{t}), \quad m = 1, 2$$
$$f_{i,m} - f_{j,m} - \phi_m(\mathcal{G}(\mathbf{p}(i) - \mathbf{p}(j))) = f^{+}_{i,j,m} - f^{-}_{i,j,m},$$
$$\forall \{\mathbf{p}(i), \mathbf{p}(j)\} \in \mathcal{N}, \quad m = 1, 2$$
$$\xi_{\mathbf{s},\mathbf{t}}, \ f^{+}_{i,j,m}, f^{-}_{i,j,m} \geq 0$$

where L is the edge point sets in a target video frame; $c(k, \mathbf{t}) = ||A\mathbf{p}(k) - \mathbf{t}||$. Since $c(k, \mathbf{t})$ is convex with respect to $\mathbf{t}$, the linear program is equivalent to the original problem without the discrete target point constraint. We use the successive convexification method to find the discrete solution. Fig. 4.12 shows one boundary tracking result. In this experiment, we randomly select 80 feature points inside the boundary of the object. We also select an initial boundary with 50 points as the contour template. The internal feature points are tracked based on the method proposed in the previous section. Because the object is not a planar object, the affine contour prediction itself is not an accurate boundary contour. With the propose method we can get a more precise contour of the object. The meshes in Fig. 4.12 show neighborhood relations of boundary points.

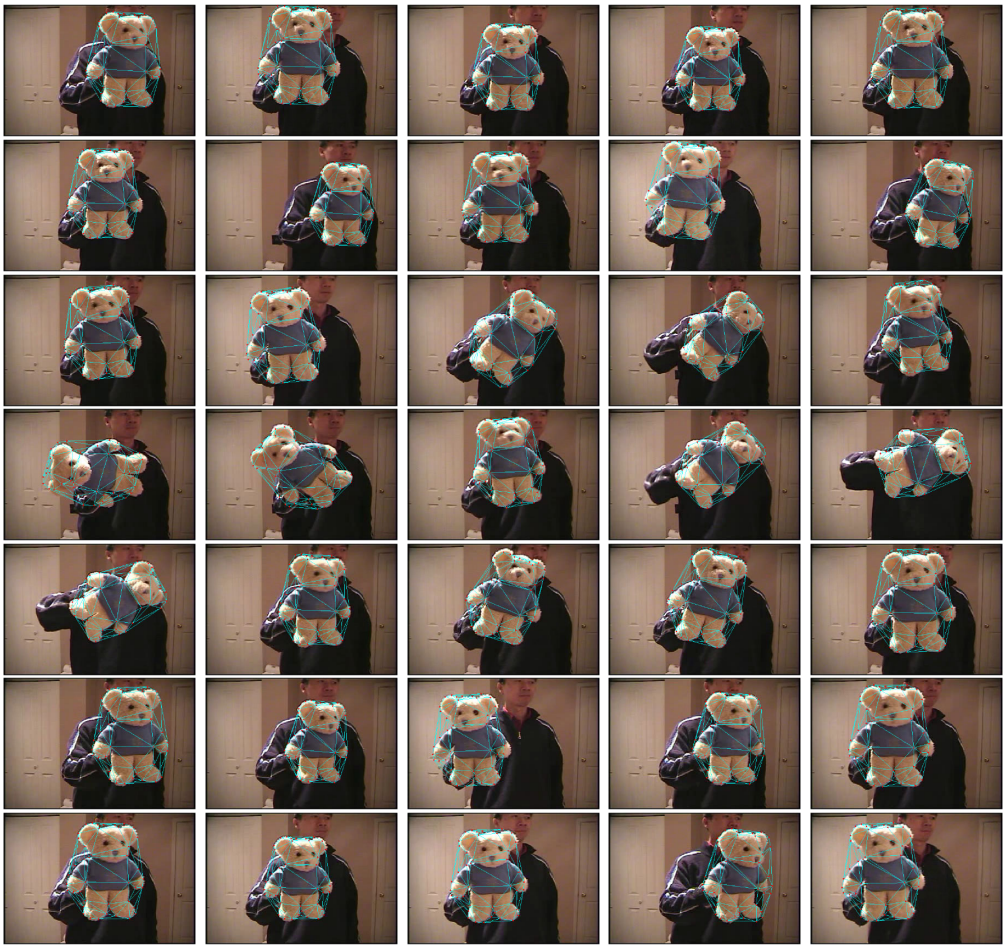


Figure 4.12: Boundary tracking. Selected frames from a 700-frame video.

4.4 Summary

In this chapter, we propose an appearance adaptive object tracking method based on successive convexification. The proposed scheme is able to track objects undergoing large rotation and scale changes. It also greatly reduces the drifting problem of most previous tracking methods. Templates are updated based on the estimation of the global transformation resulting from mean square estimation of the pointwise matching patterns, and template matching is solved by successive convexification. By choosing the best template to its corresponding matching patterns, the proposed scheme can be used to robustly track objects changing appearance dramatically, using only a few templates. Since no complex training, but only several exemplars are needed, the proposed scheme is easier to deploy than other appearance-model based schemes. Experiments show robust tracking results in cluttered environments. Convex programming method can also be used for boundary tracking. This method is more robust than traditional active contour schemes.

Chapter 5

Posture and Action Detection

Recognizing human postures and actions has been attracting a lot of interest in recent years because of its potential important applications in surveillance, multimedia database indexing and human computer interaction. In this chapter, we study posture and action detection. Our goal is to search an image database or a video sequence to locate similar postures or actions with a given exemplar image or video sequence. Most previous methods for posture or action detection are motion based, which cannot be applied to static images and also meet difficulties when a video involves rapid camera motion. Appearance based methods are more appropriate for these applications. We follow the appearance based approach and study a robust template matching based scheme for recognizing human postures in static images and general videos. The successive convexification scheme is used to match body postures in strong background clutter. Using local features, the proposed scheme is able to accurately locate and match human objects over large appearance changes. By normalizing the point-wise matching target, postures are recognized based on similarity measures between exemplars and located target objects. We further extend the deformable template matching scheme into action detection. This application is more complex and we need to match a sequence of coupled posture templates to a video sequence. We propose a successive convex relaxation method that incorporates such temporal constraints. Experiments show very promising results for the proposed scheme in recognizing and detecting human body postures and actions in images and general videos. As shown in Fig. 5.1, this chapter extends the successive convexification scheme to shape and time-space sequence matching and studies methods for posture and action detection.

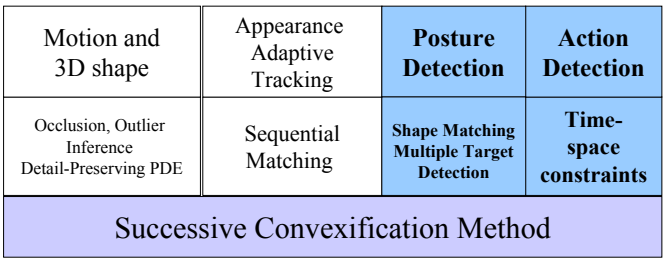


Figure 5.1: Successive convexification and extensions for posture and action detection.

5.1 Posture Recognition as a Matching Problem

First we need to define the model of human postures in images. As discussed before in the introduction, human posture can be represented as configurations of parts, e.g., head, torso, arms, legs; or, we can represent postures using smaller local features. Here, we use local features which can be extracted from images automatically. Posture recognition is thus formulated as recognizing the configuration of these local features. Feature configuration is represented as a graph. The nodes of the graph represent local features and edges represent their spatial relations. For the same posture, the shape of the graph is allowed to deform to some degree. With such a representation model, posture detection becomes a matching problem: we need to solve the problem of matching template graph to targets in clutter. After point-wise matching from a template to a target object, we can compare the similarity of template and target and carry out posture recognition.

5.1.1 Features in Posture Detection

For posture recognition problems, the features selected must be insensitive to appearance changes of human subjects. Edge map contains most of the shape information for an object, and at the same time is not very sensitive to color changes. Edge features have been widely applied in Chamfer (edge-based) matching [55] and shape context [57] matching. We found small *blocks* on distance transform image as expressive local features. To make the local features incorporate more context, we calculate log-polar transform of the distance transform image centered on the selected feature points in the target and template images. The matching cost is specified as the normalized mean absolute difference between the log-polar transform features. The log-polar transform simulates the human visual system's foveate property and puts more focus in the center view than the periphery views. This feature is

similar to the blurred edge features in [73] for object class detection.

In the following, we assume that these features are first created and then a subset is *randomly selected* on the edges of the *template*. Uniformly sampled blocks centered on *target* image edge map pixels are used for matching. The log-polar distance transform feature increases robustness in matching but nevertheless, without a robust matching scheme the matching is still very likely to fail.

5.1.2 Optimal Matching for Posture Detection

Posture matching can be thus be stated as the following optimal consistent labeling problem [59] [58]:

$$\min_{\mathbf{f}} \left\{ \sum_{\mathbf{p}\in\{\text{Template Features}\}} c(\mathbf{p},\mathbf{f_p}) + \lambda(\sum_{\{\mathbf{p},\mathbf{q}\}\in\{\text{Neighboring Pairs}\}} ||(\mathbf{f_p}-\mathbf{f_q})-(\mathbf{p}-\mathbf{q})||) \right\} \quad (5.1)$$

The symbols in the equation have similar definitions with previous chapters. Here, $c(.,.)$ is defined based on the log-polar transform features. The neighboring relation is also defined using Delaunay graph over the feature points in the template. We take $||\cdot||$ to be the L_1 norm. The above optimization formulation is sufficient for matching postures with small scale changes. For matching problems with large scaling between templates and targets, we can apply the above template matching in several different scales and choose the one with the best matching score.

For posture matching, the energy optimization problem in Eq. (5.1) is usually highly non-convex and has many local minimums. Such problems are difficult to solve with simple greedy schemes without a good initialization process. We can robustly solve the posture matching problem with the proposed successive convexification scheme. It is as robust as BP while at the same time much more efficient.

After deformable template matching, we use the following quantities to measure the difference between the template and the matching object. We follow the similarity measures of previous chapters, while modifying the definition of average matching cost M based on the log-polar features. We define measure D as the average pairwise length changes from the template to the target. An affine transform is first estimated based on the matching and then applied to the template points before calculating D. D is further normalized with respect to the average edge length of the template. Instead of using warped image difference, the average template matching cost M is defined as the average absolute difference between

the matching target features and template features. The total matching cost is simply defined as $M + \alpha D$. About 100 randomly selected feature points are needed in calculating D and M.

5.1.3 Dealing with Multiple Targets

The above method is designed to detect a single object's posture in images. If multiple objects appear in an image, the matching scheme locates the object with the posture most similar to the template posture. There are several methods we can use to extend the posture detection method into a multiple-target posture detection system. The first method is to apply the successive convex matching scheme multiple times, until the similarity measure exceeds some threshold. Each matching locates a potential object in the image. This method has high complexity and it is also hard to determine the termination threshold. Another method uses filtering method to pre-locate potential objects. And, successive convex matching is used in the detail matching and posture detection. This method is found to be more efficient and easier to implement. In the following, we give more details about the method.

We use a correlation based method [59] to locate potential objects and remove most of the clutter. The naive scheme of matching each of the possible shapes to human targets is infeasible in real applications. We make use of a composite filtering approach that matches targets using a single template. We assume there are k possible appearances of an object, represented as images $I_1, I_2, ..., I_k$ with resolution $m \times n$. I_i appears with probability p_i. We would like to use a single $m \times n$ template P to detect objects with different appearances and minimize the mean square error: $\min_P \sum_{i=1}^{k} ||P - I_i||^2$, where $|| \cdot ||$ is the Frobenius norm. By calculating the derivatives of the objective function with respect to each element of P and setting them to zero, we have $P = \sum_{i=1}^{k} p_i I_i$, i.e., using mean square error, the optimal composite template is the expectation of the appearance of an object [75]. In real implementations, $\{I_i\}$ is a large set of exemplars. To detect an object in image I, we sweep P across the image and calculate the residue errors between the template and the image at each position. In a more efficient implementation, because the template is fixed, we simply need to calculate the correlation map $J * G_\sigma - 2I * Q_\sigma$, in which $*$ is the convolution operation; each element of J is the square of the corresponding element in image I; G_σ is a Gaussian filter kernel with standard deviation σ; Q_σ is obtained by flipping P in the x and y direction and filtering by a Gaussian with standard deviation σ; typically $\sigma = 5$. To reduce

influence of clothing changes, edge maps of template images are first converted to distance transformed grayscale images and then averaged to generate the composite template. Target images are also converted to distance transformed images in composite filtering. The local valleys of the correlation map are selected as potential positions of objects. There are usually only a small number of these local minima in the correlation map, which therefore greatly reduces the searching space during further detail matching. The coarse object detection process usually also generates many false positives but this poses no problem for action detection since the final, detail matching, step will remove the false detections.

For multiple target detection in video applications, we can further use the consistency constraint to remove the inconsistent matching between video frames. As illustrated in [80], we can use the following method based on BP to do object correspondence and false object rejection. Fig. 5.2 shows a scenario of matching objects (hockey players) in two successive frames with object disappearing inference.

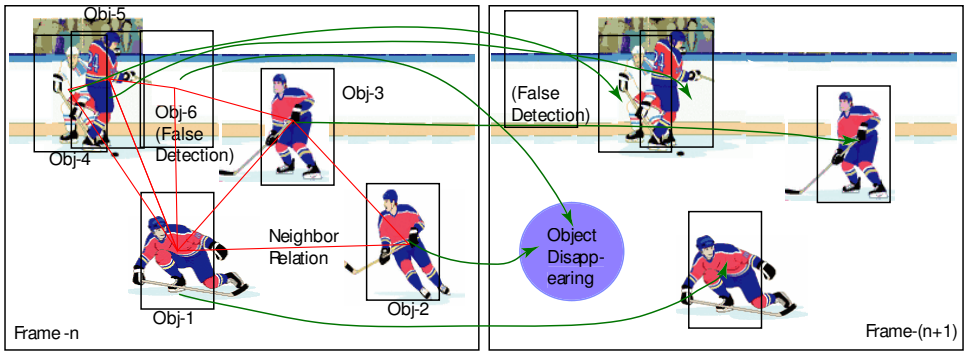


Figure 5.2: Matching objects in successive frames.

The pre-localization step based on composite filtering has high recall in detection, objects almost always appear in the detected positions in each of the frames. For inter-frame consistency check, we formulate a *consistent labeling problem*. In two successive video frames, the objects detected in the first frame are treated as sites and the objects detected in the second image are labels. The label set also contains a label to indicate an object disappearing in the second image. Each site is enforced to receive a single label and each label assignment has a given cost. The objects that do not map to the disappearing label are consistent object in these two video frames. The objects labeled as disappearing is discarded in the consistent check. We use color histogram to quantify the similarity of objects and we enforce the consistency of labeling using object's spatial relations. This is a standard consistent

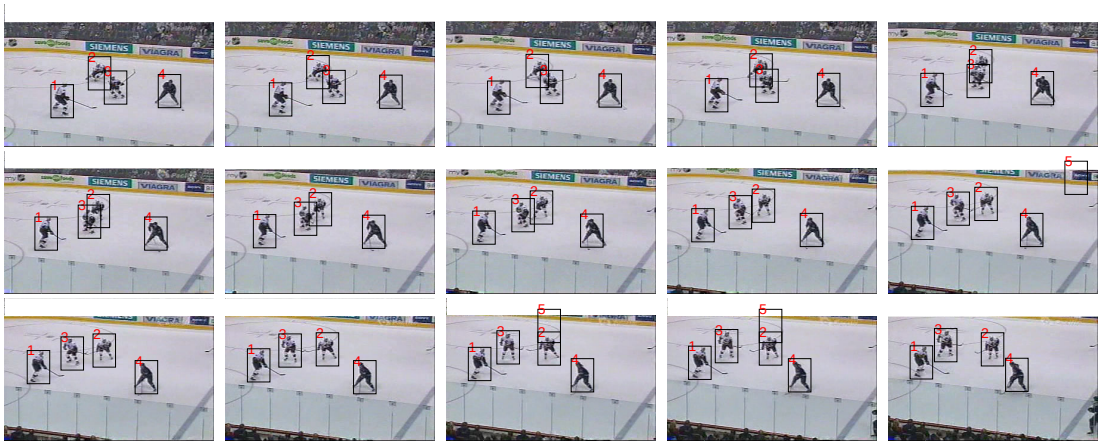


Figure 5.3: Finding and tracking hockey players by composite filtering and object correspondence in videos.

labeling problem and we use belief propagation to solve the labeling problem because the problem is usually in small scale.

Based on the filtering results, we can further conduct detail matching with successive convexification for posture detection. Besides checking the consistency of objects in two video frames, the above method also generates correspondence of potential objects in the video which can be used in action detection. One example is shown in Fig. 5.3. Fig. 5.4 shows object trajectories in a longer hockey video sequence. In Section 5.2, we study an action detection method in which no such explicit correspondence procedure is required.

5.1.4 Posture Detection in Very Large Image Database

The proposed successive convexification scheme is more efficient than previous schemes such as BP for posture detection. But it is still quite expensive for posture detection in a very large image database. In a posture clustering application [78], we need to compute the posture distance between human subjects in each image pair from an image dataset containing thousands of images. To solve the problem, shape context matching can be first used to generate a coarse matching list. In the coarse matching stage, we do not use spatial constraint. The size of shape context is also bigger than the log-polar feature in detail matching. Each shape context feature on the template finds the best match in the target image and the overall matching score is the average distance of shape context features. The detail matching based on successive convexification can be used to refine the matching result

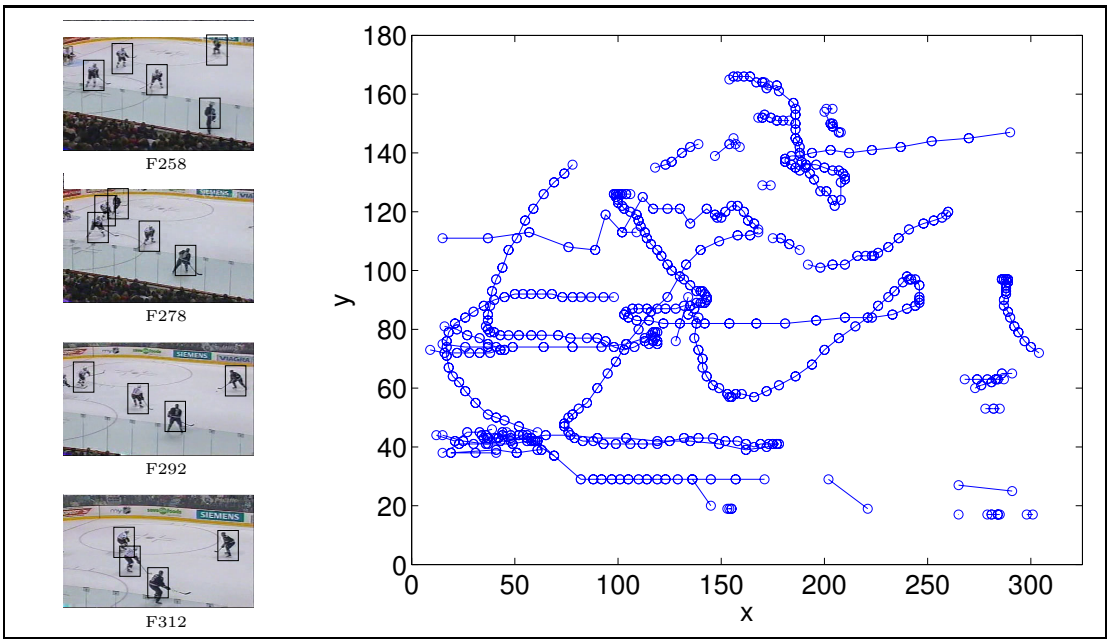
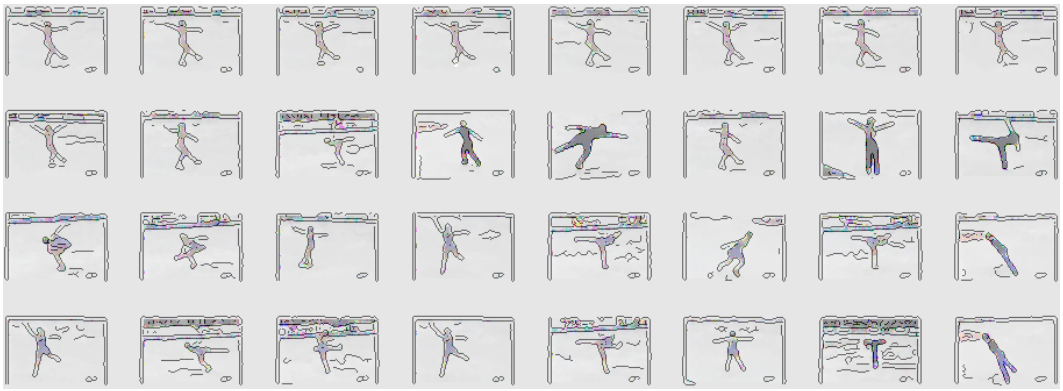


Figure 5.4: Trajectories of objects in 55 frames.

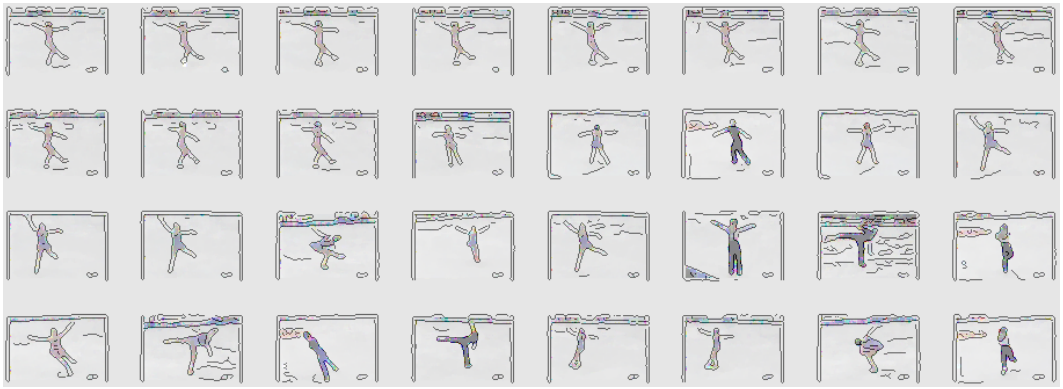
in the top n list. The deformable matching improves the matching and the distance measure between the target and template posture. These distances can further be used for clustering postures or actions as demonstrated in [78]. Figs. 5.5, 5.6, and 5.7 illustrate some sample posture matching shortlists from thousands of matching results. In each shortlist, the first image shows the template posture and the others are sorted based on their matching scores.

5.2 Action Recognition based on Time-Space Matching

In the previous section, we applied the proposed successive convexification scheme to human posture detection. For action recognition, the problem becomes more complex and we need to study methods for comparing posture template sequence with target video sequences. In this section, we study methods to detect a specific human action in an uncontrolled setting. We represent an action as a sequence of body postures with specific temporal constraints. We can then search for a given action by matching a sequence of coupled body posture templates to the video sequence. We formulate the matching problem as an energy minimization problem. The objective function is minimized such that the matching cost is low and at the same time we try to smooth the intra-frame matching and inter-frame object



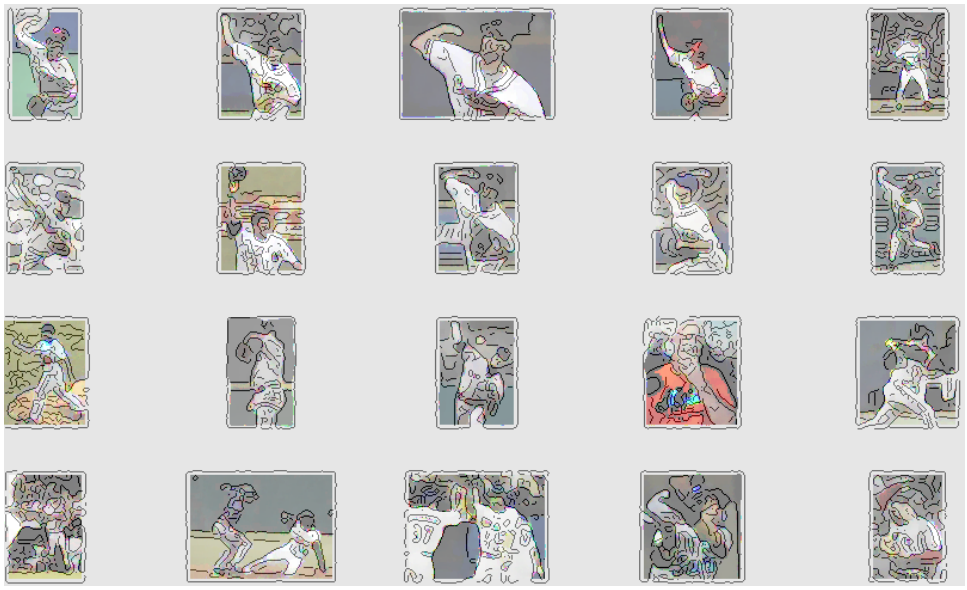
(a) Pre-sorting result



(b) Refined Result

Figure 5.5: Example shortlists of shape context pre-matching and successive convex matching refined result. The first image in each short list shows the template posture. The figure shows edge maps overlapped on low-passed color images.

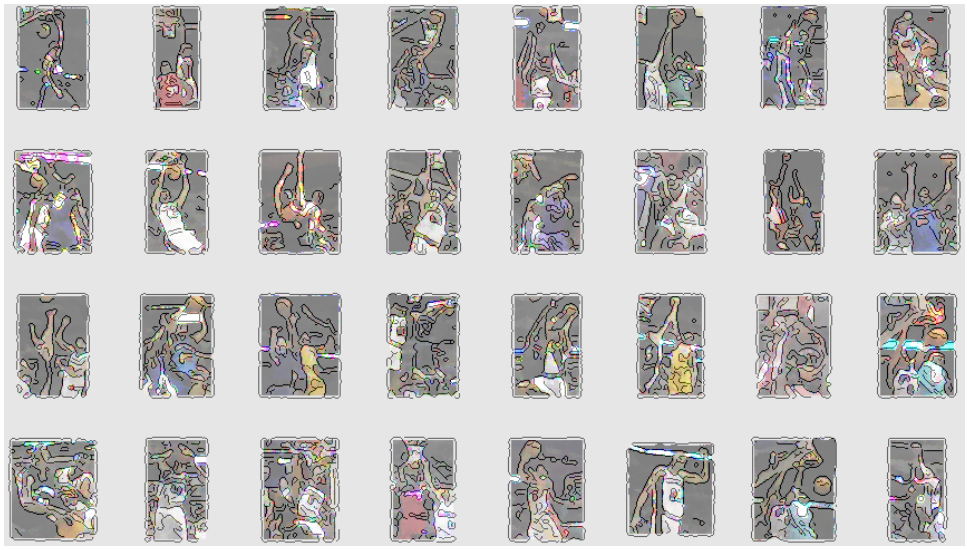
(a) Pre-sorting result



(b) Refined result

Figure 5.6: Example shortlists of shape context pre-matching and successive convex matching refined result. The first image in each shortlist shows the template posture. The figure shows edge maps overlapped on low-passed color images.

(a) Pre-sorting result



(b) Refined result

Figure 5.7: Example shortlists of shape context pre-matching and successive convex matching refined result. The first image in each show list shows the template posture. The figure shows edge maps overlapped on low-passed color images.

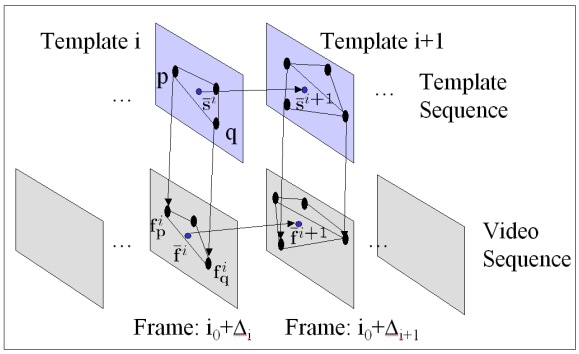


Figure 5.8: Deformable video matching.

center's relative position. A center continuity constraint is important to force the matching to stick to one object in cluttered video where multiple objects may appear. We follow the successive convexification scheme that has been applied to object matching, motion estimation and tracking, reshaping the problem so that the inter-frame constraint can be introduced. Instead of directly solving the optimal matching problem, the proposed scheme converts the optimization problem into easier convex problems and linear programming is applied to solve the sub-problems. An iterative process updates the trust region and successively improves the approximation. This convex matching scheme has many useful features: it involves only a small set of basis target points, and it is a strong approximation scheme. It is also found to be robust against strong clutter and large deformations, necessary for success of an action recognition scheme. After template to video registration, we compare the similarity of the matching targets in video with the templates by matching cost and degree of deformation.

5.2.1 Matching Templates to Video

We formulate the sequence matching problem as follows. We have n templates from a video sequence, which represent key postures of an object in some specific action. Template i is represented as a set of feature points S_i and the set of neighboring pairs $\mathcal{N}_i$. $\mathcal{N}_i$ consists of all the pairs of feature points, connected by edges in the Delaunay graph of S_i. Fig. 5.8 illustrates intra-frame and inter-frame constrained deformable video matching.

Matching a template sequence to a video can be formulated as an optimization problem.

We search for matching function $\mathbf{f}$ to minimize the following objective function:

$$\min_{\mathbf{f}} \sum_{i=1}^{n} \sum_{\mathbf{s} \in S_i} C^i(\mathbf{s}, \mathbf{f}_{\mathbf{s}}^i) + \lambda \sum_{i=1}^{n} \sum_{\{\mathbf{p},\mathbf{q}\} \in \mathcal{N}_i} d(\mathbf{f}_{\mathbf{p}}^i - \mathbf{p}, \mathbf{f}_{\mathbf{q}}^i - \mathbf{q}) + \mu \sum_{i=1}^{n-1} d(\bar{\mathbf{s}}^{(i+1)} - \bar{\mathbf{s}}^i, \bar{\mathbf{f}}^{(i+1)} - \bar{\mathbf{f}}^i)$$

Here, $C^i(\mathbf{s}, \mathbf{f}_{\mathbf{s}}^i)$ is the cost of matching feature point $\mathbf{s}$ in template i to point $\mathbf{f}_{\mathbf{s}}^i$ in a target frame; $\bar{\mathbf{f}}^i$ and $\bar{\mathbf{s}}^i$ are centers of the matching target points and template points respectively for the ith template; Distance $d(.,.)$ is a convex function. The first term in the objective function represents the cost of a specific matching configuration. The second and third terms are intra-frame and inter-frame regularization terms respectively. The coefficients λ and μ are used to control the weights of the smoothing terms. we focus on problems in which $d(.,.)$ is defined as an L_1 norm. As will be shown later, in this case, a linear programming relaxation of the problem can be constructed. To simplify the matching process, we enforce that target points for one template cannot be dispersed into several target frames. The matching frame for template i is specified as $i_0 + \Delta_i$, in which i_0 is a start frame number and Δ_i are time stamps of a template frame.

The above optimization problem is non-linear and usually non-convex, because matching cost functions $C^i(\mathbf{s}, \mathbf{t})$ are usually highly non-convex with respect to $\mathbf{t}$ in real applications. In the following, we discuss feature selection and methods to cast the non-convex optimization problem into a sequential convex programming problem, so that a robust and efficient optimization solution can be obtained.

5.2.2 Linear Programming Relaxation

The linearization process is similar to the standard consistent labeling problem in Chapter 2. The details are as follows. To linearize the matching cost term in the non-linear objective function, we select a set of *basis target points* for each feature point in a template. Then, a target point can be represented as a linear combination of these basis points, e.g, $\mathbf{f}_{\mathbf{s}}^i = \sum_{\mathbf{t} \in B_{\mathbf{s}}^i} w_{\mathbf{s},\mathbf{t}}^i \cdot \mathbf{t}$, where $\mathbf{s}$ is a feature point in template i, and $\mathcal{B}_{\mathbf{s}}^i$ is the basis target point set for $\mathbf{s}$. We will show that the "cheapest" basis set for a feature point consists of the target points corresponding to the matching cost surface's *lower convex hull* vertices. Therefore $\mathcal{B}_{\mathbf{s}}^i$ is usually much smaller than the whole target point set for feature point $\mathbf{s}$. This is a key step to speed up the algorithm. We can now represent the cost term as a linear combination of the costs of basis target points. For template i, the matching cost term can thus be represented as $\sum_{\mathbf{s} \in S_i} \sum_{\mathbf{t} \in \mathcal{B}_{\mathbf{s}}^i} w_{\mathbf{s},\mathbf{t}}^i C^i(\mathbf{s}, \mathbf{t})$. Similar to previous problems, the standard linear

programming trick of using auxiliary variables can be also used to turn L_1 terms in the objective function into linear functions. In our formulation, the summation of auxiliary variables equals the absolute value of the original term when the linear program is indeed optimized.

The complete linear program is written as:

$$\min \sum_{i=1}^{n} \sum_{\mathbf{s}\in S_i} \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}^i} w_{\mathbf{s},\mathbf{t}}^i C^i(\mathbf{s},\mathbf{t}) + \lambda \sum_{i=1}^{n} \sum_{\{\mathbf{p},\mathbf{q}\}\in\mathcal{N}_i} (x_{\mathbf{p},\mathbf{q}}^{i+} + x_{\mathbf{p},\mathbf{q}}^{i-} + y_{\mathbf{p},\mathbf{q}}^{i+} + y_{\mathbf{p},\mathbf{q}}^{i-}) + \mu \sum_{i=1}^{n-1} (u^{i+} + u^{i-} + v^{i+} + v^{i-})$$

$$\begin{aligned}
s.t. \sum_{\mathbf{t}\in\mathcal{B}_\mathbf{s}} w_{\mathbf{s},\mathbf{t}}^i &= 1, \forall \mathbf{s} \in S_i, i = 1..n \\
x_\mathbf{s}^i &= \sum_{\mathbf{t}\in B_\mathbf{s}^i} w_{\mathbf{s},\mathbf{t}}^i \cdot x(\mathbf{t}),\ y_\mathbf{s}^i = \sum_{\mathbf{t}\in B_\mathbf{s}^i} w_{\mathbf{s},\mathbf{t}}^i \cdot y(\mathbf{t}) \\
x_{\mathbf{p},\mathbf{q}}^{i+} - x_{\mathbf{p},\mathbf{q}}^{i-} &= x_\mathbf{p}^i - x(\mathbf{p}) - x_\mathbf{q}^i + x(\mathbf{q}), \\
y_{\mathbf{p},\mathbf{q}}^{i+} - y_{\mathbf{p},\mathbf{q}}^{i-} &= y_\mathbf{p}^i - y(\mathbf{p}) - y_\mathbf{q}^i + y(\mathbf{q}), \\
\forall \{p,q\} &\in \mathcal{N}_i, i = 1..n
\end{aligned}$$

$$u^{i+} - u^{i-} = \frac{1}{|S_i|} \sum_{\mathbf{s}\in S_i} [x_\mathbf{s}^i - x(\mathbf{s})] - \frac{1}{|S_{i+1}|} \sum_{\mathbf{s}\in S_{i+1}} [x_\mathbf{s}^{i+1} - x(\mathbf{s})],$$

$$\begin{aligned}
v^{i+} - v^{i-} &= \frac{1}{|S_i|} \sum_{\mathbf{s}\in S_i} [y_\mathbf{s}^i - y(\mathbf{s})] - \frac{1}{|S_{i+1}|} \sum_{\mathbf{s}\in S_{i+1}} [y_\mathbf{s}^{i+1} - y(\mathbf{s})], \\
&\quad i = 1..n-1 \\
\text{All variables} &\geq 0
\end{aligned}$$

Here we denote functions $x(\mathbf{s})$ and $y(\mathbf{s})$ as extracting the x and y component of point $\mathbf{s}$. The matching result $\mathbf{f}_\mathbf{s}^i = (x_\mathbf{s}^i, y_\mathbf{s}^i)$. It is not difficult to verify that either $x_{\mathbf{p},\mathbf{q}}^{i+}$ or $x_{\mathbf{p},\mathbf{q}}^{i-}$ (similarly $y_{\mathbf{p},\mathbf{q}}^{i+}$ or $y_{\mathbf{p},\mathbf{q}}^{i-}$, u^{i+} or u^{i-} and v^{i+} or v^{i-}) will become zero when the linear programming achieves its minimum; therefore we have $x_{\mathbf{p},\mathbf{q}}^{i+} + x_{\mathbf{p},\mathbf{q}}^{i-} = |x_\mathbf{p}^i - x(\mathbf{p}) - x_\mathbf{q}^i + x(\mathbf{q})|$, $y_{\mathbf{p},\mathbf{q}}^{i+} + y_{\mathbf{p},\mathbf{q}}^{i-} = |y_\mathbf{p}^i - y(\mathbf{p}) - y_\mathbf{q}^i + y(\mathbf{q})|$, and so on. The second and third regularization terms in the linear program objective function equal the corresponding terms in the original non-linear problem. In fact, if $\mathcal{B}_s^i$ contain all the target points $\mathcal{L}_\mathbf{s}^i$ and weights $w_{\mathbf{s},\mathbf{t}}^i$ are binary variables (0 or 1), the LP becomes an integer programming problem which exactly equals the original non-convex problem. But, integer programming is as hard as the original non-linear problem, and therefore we are most interested in the relaxed linear programming problem. This linear

program has the following properties which are very similar to the basic consistent labeling convex relaxation in Chapter 2. We briefly discuss the properties.

Property 5.1*: If* $\mathcal{B}^i_{\mathbf{s}} = \mathcal{L}^i_{\mathbf{s}}$, where $\mathcal{L}^i_{\mathbf{s}}$ is the entire label set of $\mathbf{s}$ for template i, and *the continuous extension cost function* $C^i(\mathbf{s}, \mathbf{t})$ *is convex with respect to* $\mathbf{t}$, $\forall \mathbf{s} \in S_i$, $i = 1..n$, *LP exactly solves the continuous extension of the discrete matching problem.*

In practice, the cost function $C^i(\mathbf{s}, \mathbf{t})$ is usually highly non-convex with respect to $\mathbf{t}$ for each site $\mathbf{s}$. In this case:

Property 5.2*: The linear programming formulation solves the continuous extension of the reformulated discrete matching problem, with* $C^i(\mathbf{s}, \mathbf{t})$ *replaced by its lower convex hull for each site* $\mathbf{s}$.

Property 5.3*: We need only consider the basis set* $\mathcal{B}_{\mathbf{s}}$ *comprised of the vertex coordinates of the lower convex hull of* $C^i(\mathbf{s}, \mathbf{t})$, $\forall \mathbf{s} \in S$.

We can also use only the smallest basis set in the optimization, which is one of the key steps to speed up the algorithm.

The proposed solution of the relaxation scheme also has the following structure property.

Property 5.4*: Using the simplex method, there will be at most 3 nonzero-weight basis target points for each site.*

The property can be proved similarly to those of the previous chapters. We omit the details here.

The initial basic variables can be selected in the following way:

- Only one $w^i_{\mathbf{s},\mathbf{t}}$ is selected as basic LP variable for each site $\mathbf{s}$ in template i.
- $x^i_{\mathbf{s}}$, $y^i_{\mathbf{s}}$ are basic LP variables.
- Whether $x^{i+}_{\mathbf{p},\mathbf{q}}$ or $x^{i-}_{\mathbf{p},\mathbf{q}}$, $y^{i+}_{\mathbf{p},\mathbf{q}}$ or $y^{i-}_{\mathbf{p},\mathbf{q}}$, u^{i+} or u^{i-} and v^{i+} or v^{i-} are basic variables depends on the right hand side of the constraint; if the right hand side of a constraint is greater than 0, the plus term is a basic variable, otherwise the minus term becomes the basic variable.

Property 5.4 implies that for each site the proposed LP relaxation still searches only the triangles of the lower convex hull vertices, in an efficient energy descent manner. (And note that the triangles may be degenerate.) Fig. 5.9 and Fig. 5.10 illustrates the solution procedure of the simplex method for an example two-frame video matching problem. In this

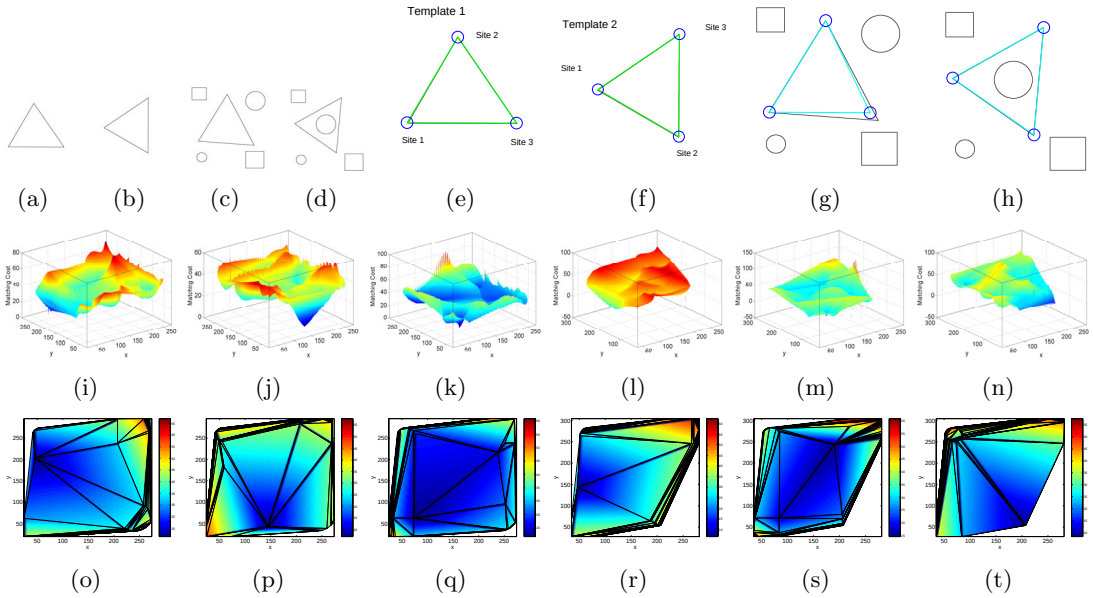
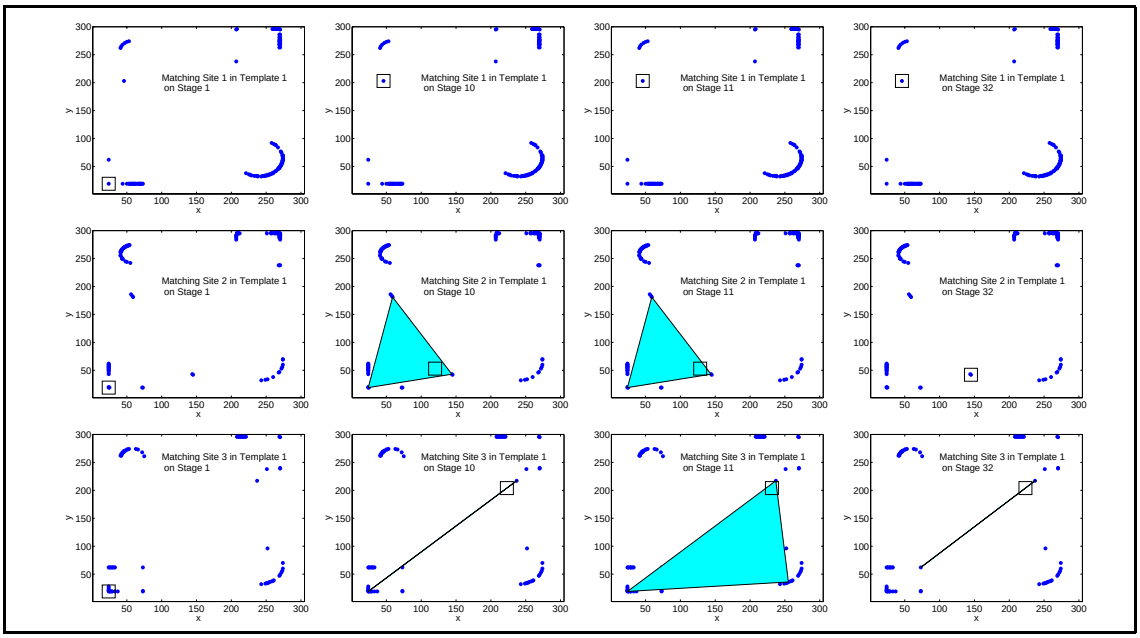
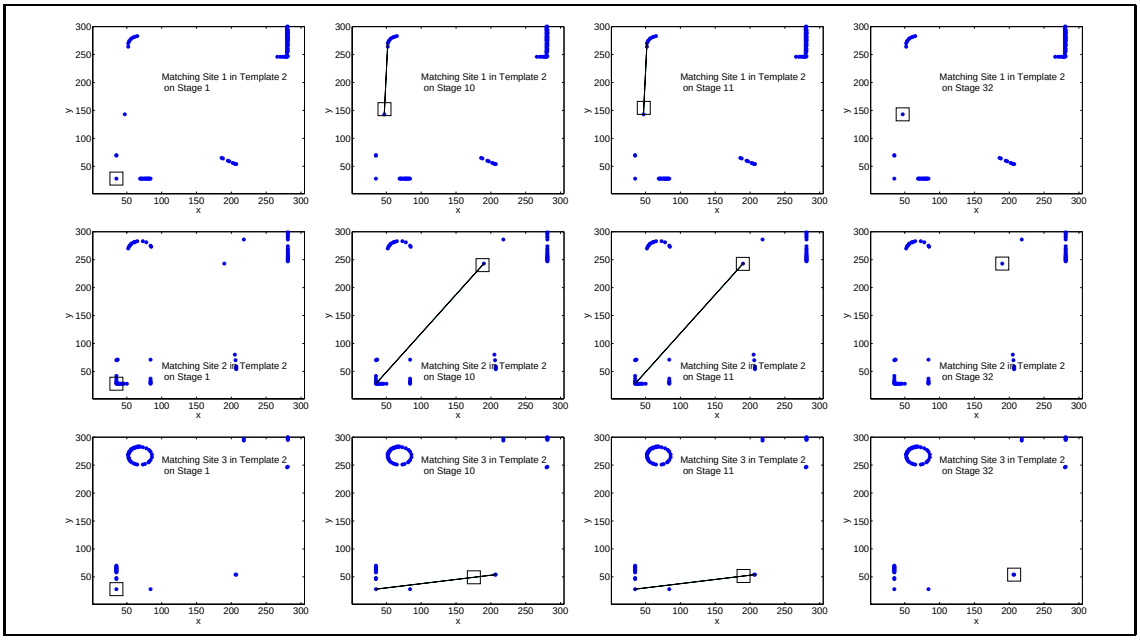


Figure 5.9: Searching process of the linear program. (a,b) Template images; (c,d) Target images; (e,f) Feature points and template graph; (g,h): Matching result; (i,j,k,l,m,n) Matching cost surfaces for each site on the template; (o,p,q,r,s,t) Convexified matching cost surfaces.

simple example, three features points are selected on the objects in Figs. 5.9(a, b) respectively and form triangular graph templates, shown in Figs. 5.9(e, f). Figs. 5.9(c, d) show the target objects in clutter. Figs. 5.9(g, h) show the matching result. Figs. 5.9 (i,j,k,l,m,n) show the matching cost surface for each of the six points on the template. Figs. 5.9 (o,p,q,r,s,t) are the lower convex hull surfaces for the respective cost surfaces. The searching process (selected from 32 stages) for each site is illustrated in this example. The block dots indicate the target points located at the coordinates of the lower convex hull vertices. The target points corresponding to the basic variables are connected by lines. The small rectangle is the weighted linear combination of the target points corresponding to the basic variables at each stage. As expected, the proposed LP only checks triangles (filled-blue) or their degenerates (lines or points) formed by basis target points. When the search terminates, the patch generated by the basic variables for each site must correspond to vertices, edges or facets of the lower convex hull for each site. As shown in this example, a single LP relaxation usually has a matching result near the target but not very accurate. We will discuss how to refine the result by successively "convexifying" the matching cost surfaces.



(a) Searching for Template 1



(b) Searching for Template 2

Figure 5.10: Searching process of the linear program. (a,b) illustrate the searching process of the linear program.

5.2.3 Successive Convexification for Video Matching

As discussed above, a single LP relaxation approximates the original problem's matching cost functions by their lower convex hulls. Because of the convexification process, many local structures are removed which could make the solution not exactly locate on the true global minimum. Successive convexification method can also be applied to this problem. Instead of one step LP relaxation, we construct linear programming problems recursively based on the previous searching result and gradually shrink the matching trust region for each site. A trust region for one site is a rectangle area in the target image. Such a scheme can effectively solve the coarse approximation problem in single step LP.

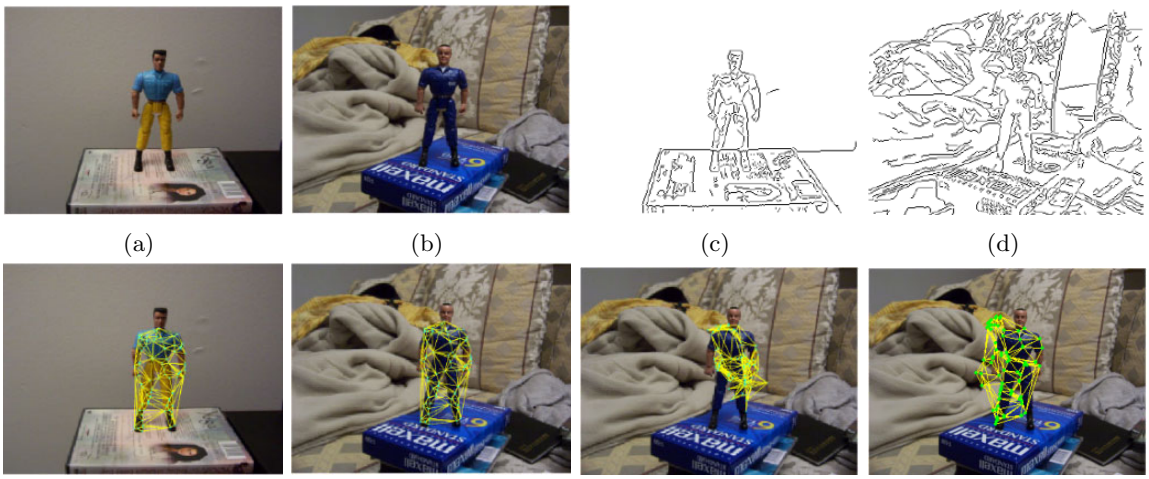
Based on the above video sequence matching scheme, we generate point-wise matching patterns from the template sequence to the target video sequence. We can proceed to compare the similarity between the target and the action template. We use similar measures in action detection with posture detection. We also use D, the average of pairwise length changes from the template to the target and average template matching cost M using the log-polar transform feature to measure the similarity of the template sequence with matched video sequence. The total matching cost is the linear combination $M + \alpha D$, where α has a typical value of 10 if image pixels are in range of 0-255.

5.3 Experimental Results

5.3.1 Finding Postures in Images

Fig. 5.11 shows an example posture matching result with successive convex matching. Log-polar features are used in matching. The proposed scheme correctly matches the object in strong clutter. ICM fails to match the object. Because of the complexity issue, searching range of Graph Cut is constrained in a window of [-50,50] in both x and y directions. The matching result of Graph Cut is also not accurate.

Finding postures in video sequences using exemplar postures is a very useful application. We first test the method with a "yoga" sequence, which is about 30 mins long. We choose three different posture exemplars from another section of the video. By specifying the region of interest, graph templates are automatically generated from the exemplars. Each template is then compared with video frames in the test video. One of every 8 frames are used in matching. The shortlists based on their matching scores are shown in Figs. 5.12 (b, c, d).



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

(e) (f) (g) (h)

Figure 5.11: (a): Template image; (b): Target image; (c): Edge map of template image; (d): Edge map of target image; (e): Template mesh; (f): Matching result of SC-LP; (g): ICM matching result; (h): GC matching result.

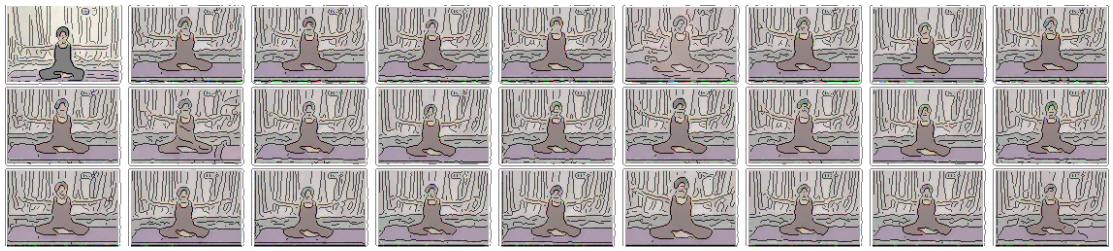
The template is shown as the first image in each shortlist. The Recall-Precision curves are displayed in Fig. 5.16 (a). As shown in Fig. 5.16 (a), the proposed posture detection scheme performs well in this experiment.

Fig. 5.13 illustrates the performance of the proposed scheme in matching objects with *large appearance differences*. We use a flexible toy as the template object and search in video sequences for similar postures of actual human bodies. Two sequences are used in testing: the first, shown in Fig. 5.13, has 500 frames and the other has 1,000 frames. There are fewer than 10% true targets in the video sequence. The vertical and horizontal edges in the background are very similar to the edge features on human bodies, and this presents a major challenge for object location and matching. The shortlists of matching results are shown in Figs. 5.13 (b, d), and Recall-Precision curves are shown in Fig. 5.16 (b).

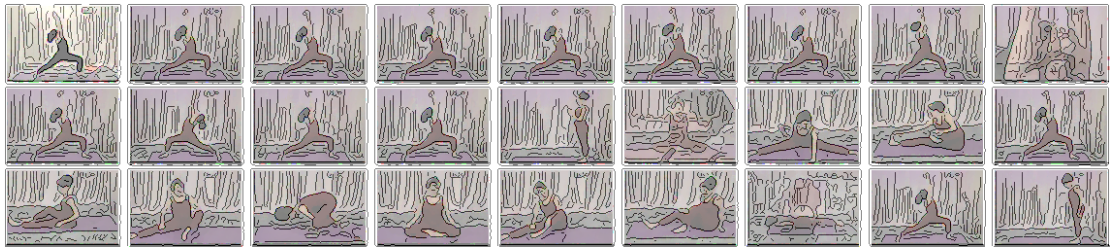
In another experiment, we search a figure skating sequence about 30 mins long. The figure skating program contains 5 skaters, with quite different clothing. The audience in the scene presents strong background clutter, which may cause problems for most matching algorithms. The sampling rate for the video is 1 frame/second. Figs. 5.14 (a), (b) and (c) show shortlists for three different postures. Images are sorted based on the matching scores. The templates are shown as the first image in each shortlist. The proposed scheme successfully detects similar postures for different people with very different clothing. The

(a) Sample frames from video

(b) Shortlist of matching for Yoga posture 1



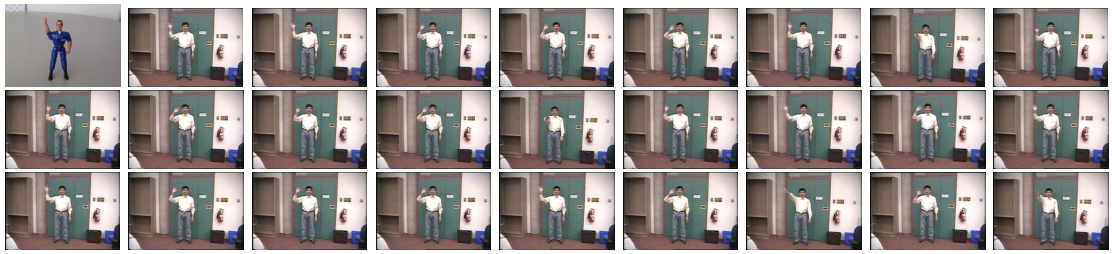
(c) Shortlist of matching for Yoga posture 2



(d) Shortlist of matching for Yoga posture 3

Figure 5.12: Posture detection in yoga sequence. The first image in each shortlist shows the template posture. The figure shows edge maps overlapped on low-passed color images.

(a) Sample frames from video 1



(b) Top 26 matches for video 1

(c) Sample frames from video 2

(d) Top 26 matches for video 2

Figure 5.13: Matching human postures using flexible toy object template. The first image in each shortlist shows the template posture.

Recall-Precision curves are shown in Fig. 5.16 (c).

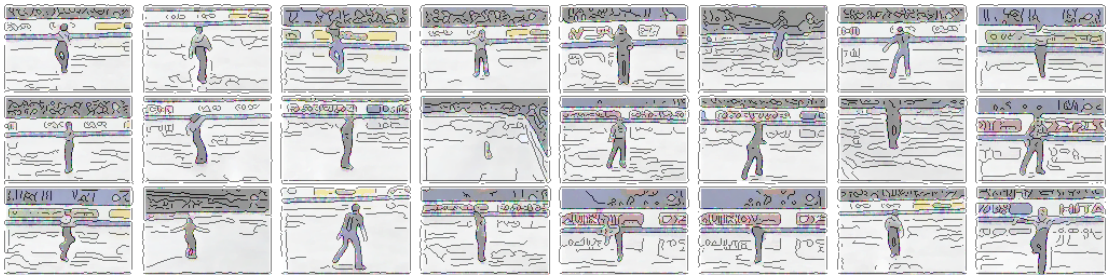
In the previous experiments, we search for postures in videos that contain a single object in each video frame. In this experiment, we consider posture recognition for videos that may contain multiple objects in each frame. Hockey is a fast paced game, with fast player movements and camera motion. Detecting activities of hockey players is an interesting and challenging application. The background audience and patterns on the ice also make posture recognition a hard problem. To deal with multiple targets in images, we apply composite filtering first. The composite template is constructed with 200 randomly selected hockey players images. To reduce the influence of clothing, these sample images are first converted to distance transformed images and then used to construct composite template. For each input video frame, local valleys in the residue image of composite filtering are potential object centers. Rectangular image patches centered on these object centers are cut from each video frame and forwarded to linear programming detail matching to compare their similarity with the posture template. Fig. 5.15 (a) shows the shortlist of searching for a shooting action in a 1,000-frame video. Two instances of the shooting action are successfully detected at the top of the shortlist. Fig. 5.15 (b) shows another posture matching result, for a 1,000-frame video with another posture template. The shortlist of video frames and hockey players are shown, based on the matching scores. The matching score for a video frame is defined as the smallest object matching score in the frame. The Recall-Precision curves are shown in Fig. 5.16 (d).

In summary, the proposed successive convexification posture detection scheme can be used to detect specific human postures in clutter backgrounds with large object distortions. It can also be used in applications in which multiple objects are involved. As shown in the Recall-Precision curves in Fig. 5.16, the overall detection rate at the equal-recall-precision point is in the range of 60%–90%.

We also conduct experiments to compare the proposed posture detection method with Chamfer matching. Experiments showed that Chamfer matching cannot deal with strong clutter and large deformation and for posture detection its result is much worse than the proposed method. Fig. 5.17 shows a Chamfer matching result using the same testing data as Fig. 5.14 (c)

(a) Top 23 matches for figure skating posture 1. The first image is the exemplar.

(b) Top 23 matches for figure skating posture 2. The first image is the exemplar.



(c) Top 23 matches for figure skating posture 3. The first image is the exemplar.

Figure 5.14: Posture detection with figure skating sequence. The figure shows edge maps overlapped on low-passed color images.

(a) Locating shooting posture in video with exemplar 1

(b) Locating postures in video with exemplar 2

Figure 5.15: Finding postures in hockey.

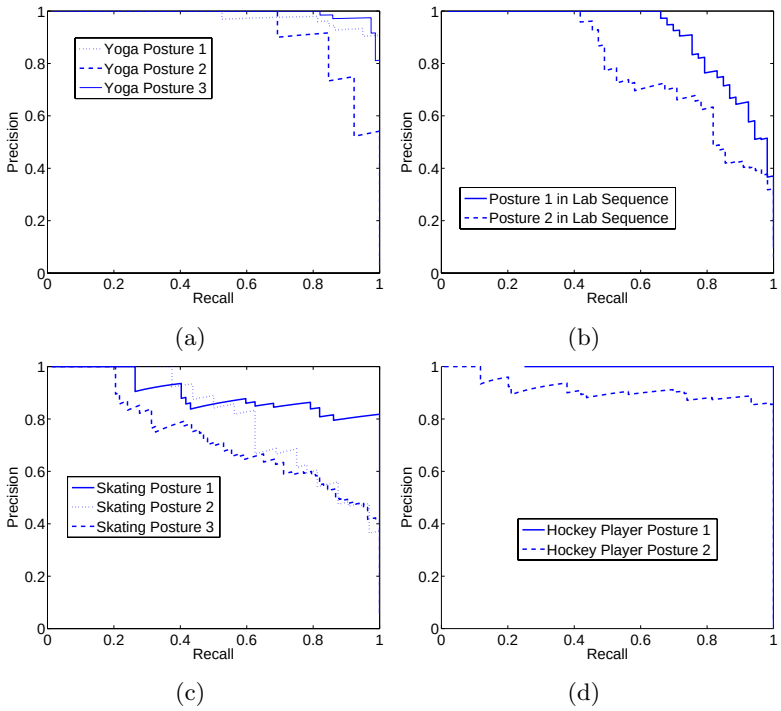


Figure 5.16: Recall-Precision curves.

5.3.2 Finding Actions in Videos

Matching Random Sequences

In the first experiment, we test the proposed scheme with synthetic images. In each experiment, three coupled random template images are generated. Each template image contains 50 random points in a 128×128 image. The target images contain a randomly shifted and perturbed version of the data points in 256×256 images. The perturbation is uniformly disturbed in two settings: 0-5 and 0-10. The center of the two templates are also randomly perturbed in the range 0-5. We use the log-polar transform feature of the distance transform image in all our experiments. We compare our result with a greedy searching scheme. Other standard schemes, such as BP, cannot be easily extended to solve the minimization problem in this chapter. Instead we use BP to match each image pair separately as a benchmark in comparison. The Graph Cut, designed mainly for stereo matching and motion estimation, is not included in the comparison. Each experiment is repeated in a deformation and clutter setting over 100 trials. Fig. 5.19 shows the average matching error distribution in different assumed-error regions. When both the noise level and distortion level are low,

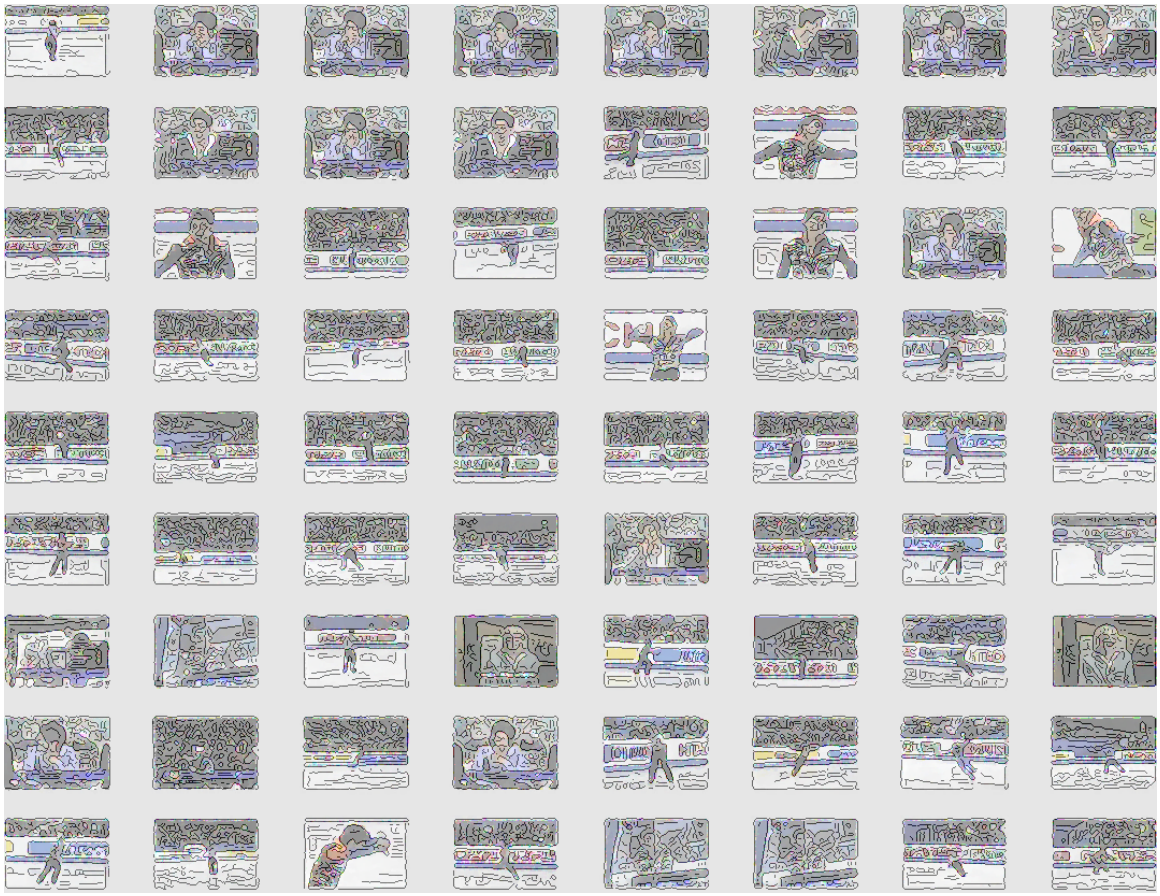


Figure 5.17: Chamfer matching result for figure skating. The first image in the shortlist shows the template posture. The figure shows edge maps overlapped on low-passed color images.

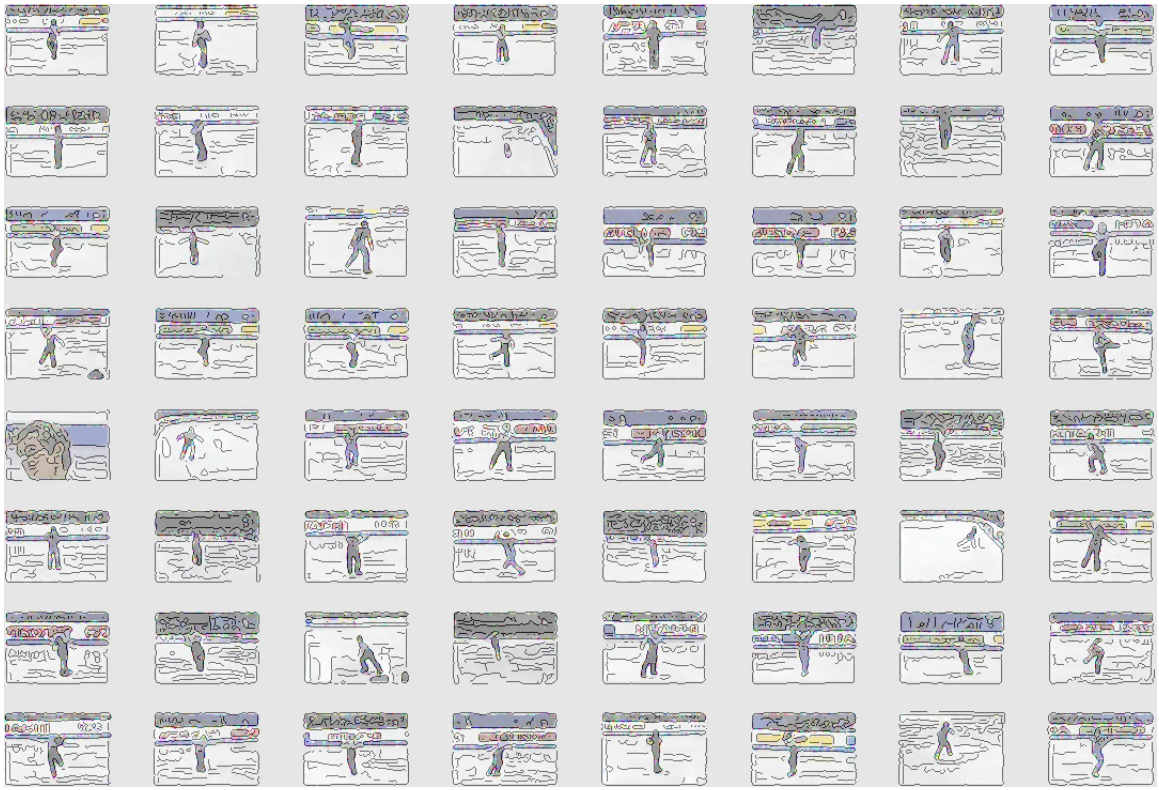


Figure 5.18: SC-LP posture detection result for figure skating. The first image in the shortlist shows the template posture. The figure shows edge maps overlapped on low-passed color images.

the greedy scheme has comparable performance. Since there is one single target in each image, BP has similar performance as the proposed scheme for low deformation setting experiments. Greedy scheme's performance degrades rapidly when the levels of noise and distortion increase. In high noise density and large deformation cases, the proposed scheme greatly outperforms the greedy scheme. It is also better than a baseline BP scheme for large distortion cases. One iteration of linear programming in this experiment takes about 0.3 seconds in a 2.6GHz PC. The typical number of iterations is 3. Fig. 5.20 shows comparison results of matching random sequences in a different outlier pattern setting which introduces an extra duplicated and perturbed object into the second target frame. For BP and greedy scheme, matching error for the second template frame is the smaller one of matching either of the two objects in the target frame. In this case, the proposed sequence matching scheme yields much better results.

Detecting Human Actions in Video

Fig. 5.21 shows an example of matching for cluttered images with the proposed sequence matching scheme, the greedy scheme, Chamfer matching and the BP matching. Chamfer matching and BP match each single image pair separately. The proposed scheme still works well in cluttered images, while the greedy scheme, Chamfer matching and BP fail to locate the target. BP is also about 100 times slower.

We test the proposed matching scheme with a dataset [83] which includes six gestures. We select templates with two key postures from the first video clip in each category and then randomly select 15 other video clips in each class for testing (The video clips with mirror action are not included). Fig. 5.22 shows examples of the two-key-frame templates in the first row of each sub-figure. We select regions in the two-frame templates for each of the six actions. Graph templates are automatically generated using randomly selected edge points in the region of interest. The templates are then used to compare with each testing video clip at each time instant using the proposed matching scheme and the minimal matching cost is used as the score for a clip. Three time scales 0.75, 1 and 1.25 are used in searching. Spatial scale is fixed in the experiment. A video clip is classified as an action if the corresponding key-posture templates generate the lowest match score among six actions. Fig. 5.22 shows some matching examples and Table 5.1 shows the detection confusion matrix: the method performs very well.

We further conducted experiments to search for a specific gesture in video. In these test

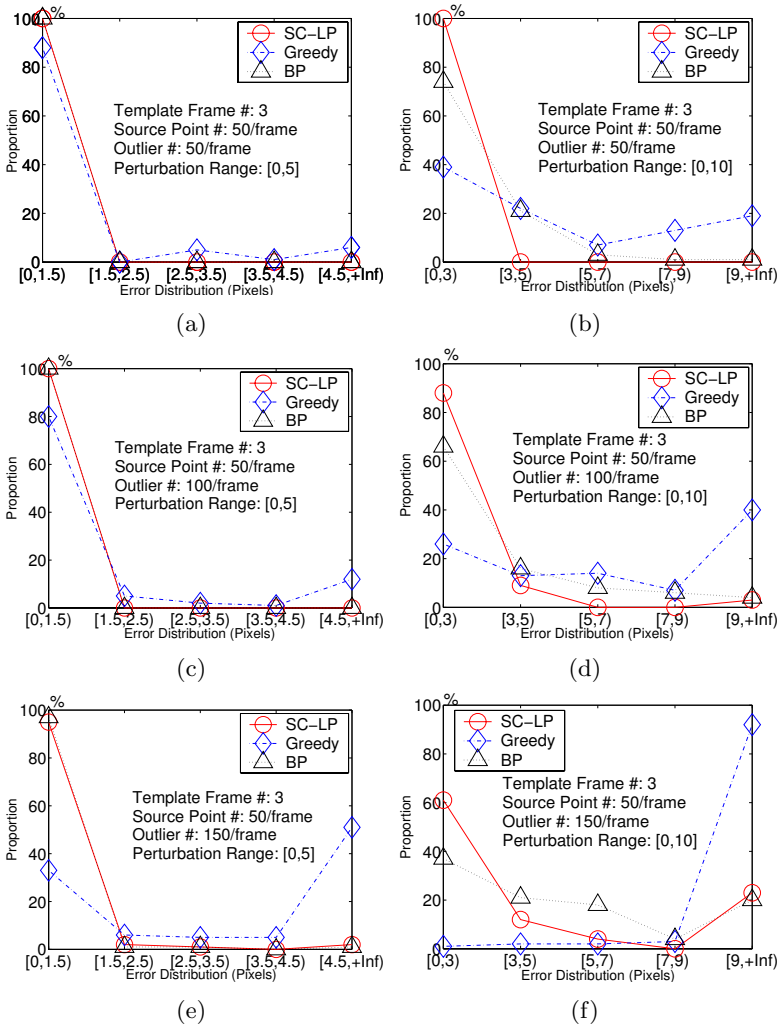


Figure 5.19: Random sequence matching.

Table 5.1: Confusion matrix for 15 random selected video clips in each action class.

	box	clap	jog	walk	run	wave
box	14	1	0	0	0	0
clap	2	13	0	0	0	0
jog	0	0	14	0	1	0
walk	2	0	0	12	1	0
run	3	1	0	0	11	0
wave	2	1	0	0	0	12

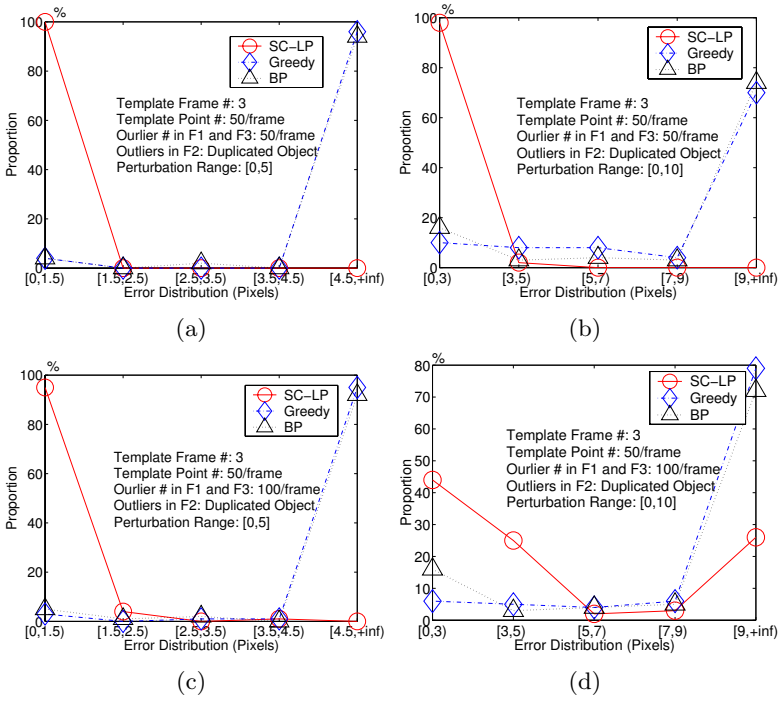


Figure 5.20: Matching sequences with multiple objects.

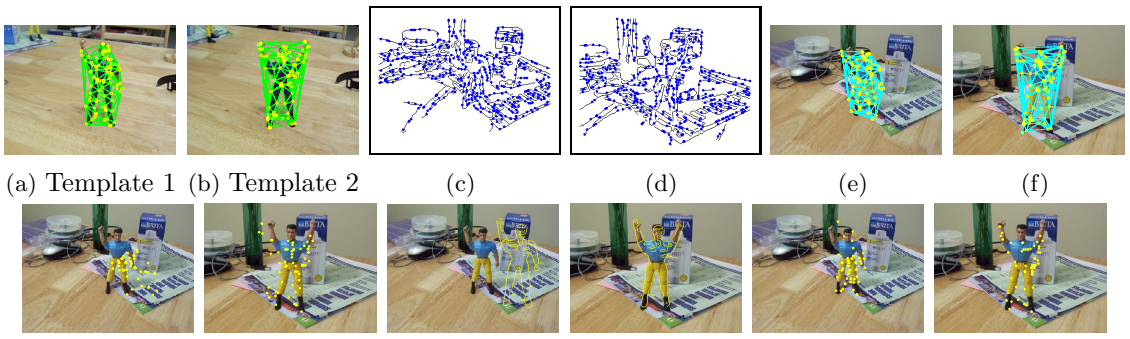


Figure 5.21: Matching flexible objects. (a, b) Templates; (c, d) Target image edge maps and feature points; (e, f) Matching with the proposed scheme; (g, h) Matching with greedy scheme; (i, j) Chamfer matching for each image pair; (k, l) Matching with BP for each image pair.

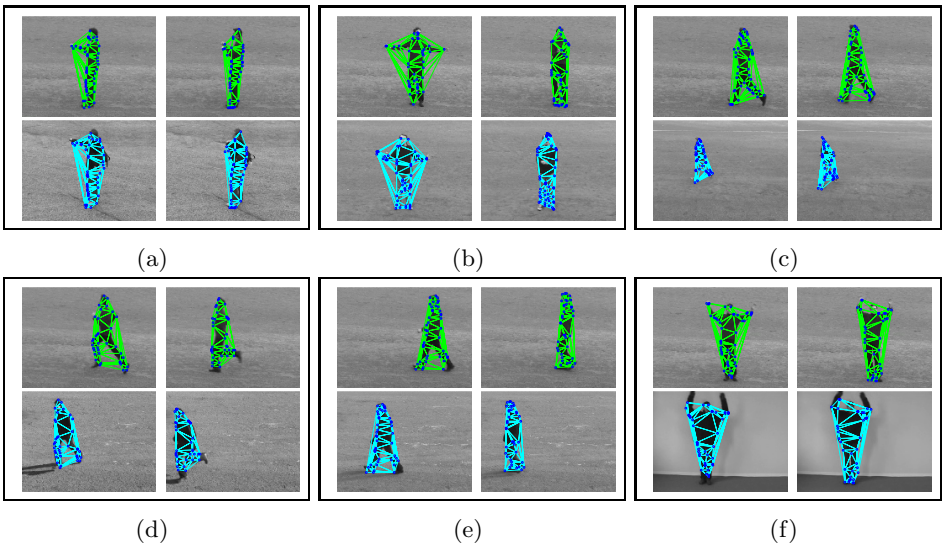
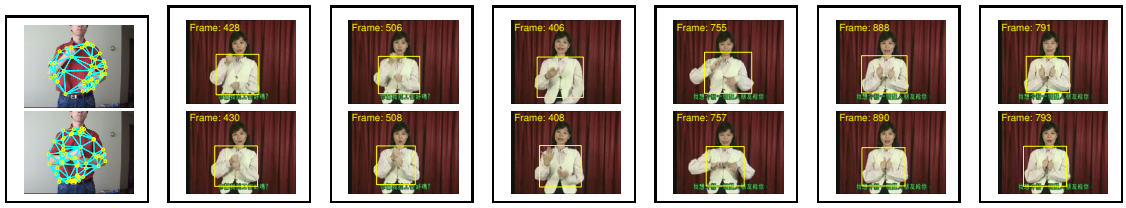


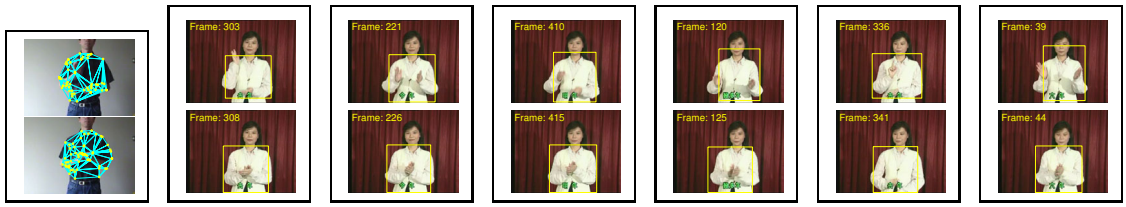
Figure 5.22: Matching Examples. In (a,b,c,d,e,f) top rows are templates and bottom rows are matching results. Action dataset ©Ivan Laptev, 2006, by permission.

videos, a specific action only appears a few times. Target objects also have large deformation with respect to the templates. The templates we use have roughly the same scale as the testing sequence. The template sequence is swept along the time axis with a step of one frame, and for each instant we match video frames with the templates. We first applied the matching scheme to detect specific sign language gestures. Sign language is challenging because of the very subtle differences. Fig. 5.23 shows a searching result for the gesture "work" in a 1000-frame video. The template sequence is generated from a different subject. The two gestures in the video are successfully located in the top two rank positions of the shortlist. Fig. 5.24 shows a searching result for the gesture "year" in a 1000-frame video. The starting and ending frames of actions in video are ranked based on their matching score. Five appearances of the gesture are located in top 6 of the shortlist. One false detection is inserted at rank 5. Fig. 5.25 and Fig. 5.26 show experiments to locate two actions, kneeling and hand-waving, in indoor video sequences of 800 and 500 frames respectively. The two-frame templates are from videos of another subject in different environments. The videos are taken indoors and contain many bar structures which are very similar to human limbs. The proposed scheme finds all the 2 kneeling actions in the test video in the top two of the shortlist; and all the 11 waving hand actions in the top 13 ranks. The result of searching for a "throwing" action in a 2500-frame baseball sequence is shown in [79]. The object occupies



(a) Templates (b) Cost:41.21 (c) Cost:41:28 (d) Cost:41.55 (e) Cost:41.75 (f) Cost:41.85 (g) Cost:41.87

Figure 5.23: Searching gesture "work" in a 1000-frame sign language sequence. (a) Templates; (b..h) Top 6 matches of the shortlist.



(a) Templates (b) Cost:43.9 (c) Cost:45.5 (d) Cost:46.0 (e) Cost:46.2 (f) Cost:46.4 (g) Cost:46.5

Figure 5.24: Searching gesture "year" in a 1000-frame sign language sequence. (a) Templates; (b..h) Top 6 matches of the shortlist.

very small part of the video. There is large deformation and strong background clutter. Closely interlaced matching results are merged and our method finds all the 4 appearances of the action at the top of the shortlist. We found that false detection in our experiments is mainly due to similar structures in the background near the subject. Prefiltering or segmentation operations to partially remove the background clutter can further increase the robustness of detection.

5.4 Summary

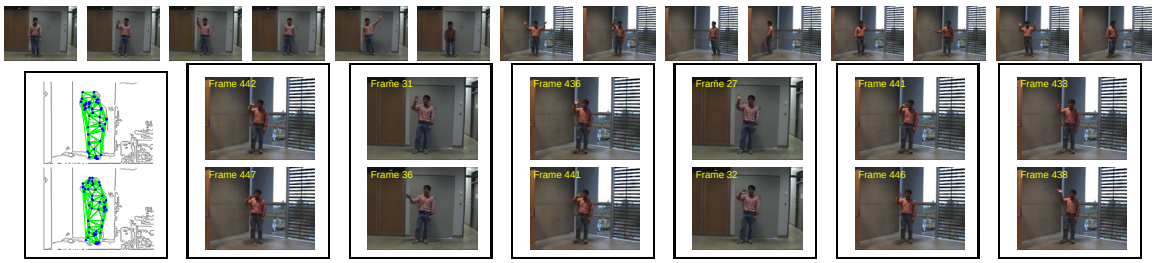
We have set out a novel linear programming method using multiple-step convexification for human body posture detection in images and videos. The method is more efficient and effective than schemes such as Belief Propagation methods for the object matching problem when a large searching range is involved. It can also solve problems for which other schemes fail. Experiments show very promising results for human body posture detection in cluttered environments.

We present a successive convex programming scheme to match video sequences using intra-frame and inter-frame constrained local features. By convexifying the optimization

(a) Templates (b) Cost:34.0 (c) Cost:34.4 (d) Cost:34.5 (e) Cost:34.6 (f) Cost:34.8 (g) Cost:34.9

(h) Cost:35.1 (i) Cost:35.2 (j) Cost:35.3 (k) Cost:35.5 (l) Cost:35.6 (m) Cost:35.7 (n) Cost:35.7

Figure 5.25: Searching "kneeling" in a 800-frame indoor sequence. (a) Templates; (b..n) Top 13 matches of the shortlist.



(a) Templates (b) Cost:40.00 (c) Cost:40.09 (d) Cost:40.09 (e) Cost:40.18 (f) Cost:40.33 (g) Cost:40.33

(h) Cost:40.37 (i) Cost:40.45 (j) Cost:40.51 (k) Cost:40.54 (l) Cost:40.55 (m) Cost:40.62 (n) Cost:40.73

Figure 5.26: Searching "right hand waving" in a 500-frame indoor sequence. (a) Templates; (b..n) Top 13 matches of the shortlist.

problem sequentially with an efficient linear programming scheme which can be globally optimized in each step, and gradually shrinking the trust region, the proposed method is more robust than previous matching schemes. The matching scheme has unique features in searching: it involves a very small number of basis points and thus can be applied to problems that involve large number of target points. The proposed scheme has been successfully applied to locating specific actions in video sequences. Because the template deforms, this scheme can deal with large distortions between the template and the target object.

Chapter 6

Conclusion

In this thesis, we have devised a novel successive convexification scheme to solve consistent labeling problems with convex regularization terms. Many computer vision problems such as object matching, motion estimation, tracking, activity recognition and many others can be formulated as such problems. These problems usually have very large label sets, for which traditional schemes cannot yield satisfactory results efficiently. With the proposed scheme, the size of the successive convex problem is largely decoupled from the that of the target label set. The proposed scheme can thus be used to solve very large label set problems more efficiently than previous methods.

Successive convexification has been successfully applied to several important applications, and it yields very promising results. We have devised successive convexification schemes for object matching and localization, large scale motion with occlusion/outlier inference, appearance adaptive object tracking, and detecting human postures and actions in clutter from static images and videos.

6.1 Contributions of the Thesis

The contributions of this thesis are detailed below:

1. **Successive Convexification for Consistent Labeling**

   Successive convexification [14] [60] [79] approximates the labeling cost surfaces for each source site with their lower convex hulls and constructs convex programming based on a small number of basis labels. For an n-D labeling problem, basis labels correspond to

the $(n+1)$-D lower convex hull vertices. The successive relaxation scheme shrinks the trust regions systematically and converts consistent labeling into a sequence of much easier convex programming problems. The proposed scheme can be applied to very large label set problems. For problems that have an L_1 regularization term, successive convexification linear programming method has only an average logarithm complexity with respect to the number of labels. Successive convexification is quite general and can be applied to any convex regularization labeling problems.

2. **Extensions and Applications**

   Successive convexification is shown to be a general scheme that we have been able to apply to many applications in vision.

   - **Object Matching**

     Successive convexification has been applied to object matching [81]. It can be used to search over the range of a whole target image, involving tens of thousands of feature points. Experiments confirm that the proposed scheme is much more robust than greedy schemes. It is also much faster than other robust matching schemes such as BP.

   - **Motion Estimation**

     The successive convexification scheme has been applied to large scale motion estimation with outlier/occlusion inference [14]. The proposed scheme can be used to estimate motion as well as the occlusion map in a searching window as large as hundreds of pixels in both the x and y directions. Experiments show that the successive convex method yields the smallest motion estimation errors compared to other matching schemes such as ICM, BP and Graph Cut for large scale motion estimation using ground truth testing data. Together with a newly devised detail preserving PDE method, we are able to efficiently estimate dense motion maps. The proposed motion estimation scheme has also be applied to reconstructing the 3-D shape of surfaces, combining with a max-flow scheme.

   - **Appearance Adaptive Tracking**

We devised a robust appearance adaptive object tracking method based on successive convexification [60]. Object tracking is formulated as a sequential matching problem in which we find point-wise correspondence from the key frame templates to targets in a video. By choosing the best matching template, the proposed scheme can be used to robustly track objects even if they change appearance dramatically, using only a few templates. The proposed scheme can also deal with large scale changes and rotations and robustly track objects over thousand of frames. Since no training, but only several exemplars are needed, this method is easier to deploy than other appearance-model based schemes. The proposed scheme does not rely on distinguished features and can be used to track objects with little texture. We also propose successive convex programming schemes to do boundary tracking.

- **Posture and Action Detection**

  The proposed successive convexification scheme is well suited to posture detection [59] [58]. It can be used to generate point to point correspondence from an exemplar to target objects, even in clutter. Successive convexification can also be combined with other methods such as composite filtering and can be applied to multiple object posture detection [80]. We have also devised a successive convex programming scheme to match video sequences using intra-frame and inter-frame constrained local features [79]. By convexifying the optimization problem sequentially with an efficient convex programming scheme which can be globally optimized in each step, and gradually shrinking the trust region, the proposed method is more robust than previous schemes. Because the template deforms, this scheme can deal with large distortions between the template and the target object. It has been successfully applied to locating specific actions in video sequences. Successive convexification has also been applied to posture clustering applications [78].

6.2 Future Work

1. One future research topic is to further study how to apply the proposed successive convex programming scheme to non-convex regularization term labeling problems. We have studied a simple approximation method: we successively convexify both the

labeling cost term and the regularization term when shrinking the trust regions. After we replace the non-convex function in the regularization term with an approximated convex function, we can thus apply the successive convexification method. As we shrink the trust regions, the possible difference of labels assigned to two nearby sites are also narrowed down. Therefore, we can replace the non-convex function in the smaller region with a better approximated convex function. This will result in a sequence of convex programming which would possibly behave more robustly than directly solving the non-convex regularization problem. We would like to further study whether we can avoid the approximation process when applying successive convexification.

2. The proposed scheme is very efficient with respect to the number of labels. However, the current scheme could still become quite complex as the number of sites goes up. Currently, we use the coarse-to-fine strategy to solve this problem. We start from a sparser site set and solve the smaller labeling problem with the successive convex programming scheme. We then upgrade the result into a denser set with local searching methods. As a future project, we would like to increase the efficiency of the scheme with respect to the number of sites so that the convex programming scheme can be applied directly.

3. We also would like to study more efficient approximation schemes to simplify the label set. In theory, lower convex hull vertices correspond to the most compact basis label set. In practice, for most real problems, the label set can be further simplified. The lower convex hull sometimes contains too many details at some unimportant places, for example, the boundary regions at the early stages of convex programming. We therefore can further improve efficiency by removing these unimportant structures without sacrificing quality. Approximate lower convex hull schemes that have much less complexity without adversely affecting the solution is also another a research topic.

4. We also would like to study the application of successive convexification in more applications and to improve its efficiency so that real-time performance is achieved. Currently, the processing time for applications such as activity recognition using successive convexification is still far from real-time. Apart from studying more efficient algorithms to increase the efficiency, we would like to study hardware accelerated schemes such as parallel processing structures to improve the response time. Many parts of the labeling process using successive convexification can be parallelized. For

one, the feature pre-matching process is amenable to such a strategy. Convex programming techniques such as linear programming pivoting can also be parallelized.

5. For the specific applications in the thesis, there is still a lot of room for improvement. For object tracking, we currently use a very simple dynamic model. Such a choice is mainly because the successive convexification scheme is robust enough to overcome frequent errors in the dynamic model. More complex dynamic models will certainly improve the performance. For posture and action detection, we can devise better features so as to reduce the load on matching, etc.

Bibliography

[1] W. Takahashi, "A convexity in metric space and nonexpansive mappings I", Kodai Math. Sem. Rep., 22 (1970), 142-149.

[2] P.F. Felzenszwalb, "Representation and detection of deformable shapes", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), vol.1, pp.102-108, 2003.

[3] J. Besag, "On the statistical analysis of dirty pictures", J. R. Statis. Soc. Lond. B, 1986, vol.48, pp. 259-302.

[4] H. Ishikawa, "Global Optimization using Embedded Graphs", Ph.D. Dissertation. May 2000, NYU.

[5] Y. Weiss and W.T. Freeman. "On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs", IEEE Trans. on Information Theory, 47(2):723-735, 2001.

[6] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient belief propagation for early vision", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), vol.1, pp.261-268, 2004.

[7] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.23, pp.1222-1239, 2001.

[8] V. Kolmogorov and R. Zabih. "What energy functions can be minimized via graph cuts?", European Conference on Computer Vision (ECCV'02), pp. III:65-81., 2002.

[9] J.M. Coughlan and S.J. Ferreira, "Finding deformable shapes using loopy belief propagation", European Conference on Computer Vision (ECCV'02), vol.3, pp.453-468, 2002.

[10] H. Chui and A. Rangarajan, "A new algorithm for non-rigid point matching", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), vol.2, pp.44-51, 2000.

[11] A. Blake and A. Zisserman, Visual Reconstruction. Cambridge, Mass.: MIT Press, 1987.

[12] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching", IEEE Trans. on PAMI, vol.18, no.4, pp.377-388, 1996.

[13] C. Chekuri, S. Khanna, J. Naor, and L. Zosin, "Approximation algorithms for the metric labeling problem via a new linear programming formulation", Symp. on Discrete Algs. (SODA'01) pp.109-118, 2001.

[14] H. Jiang, Z.N. Li, and M.S. Drew, "Optimizing motion estimation with linear programming and detail-preserving variational method", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), vol.1, pp.738-745, 2004.

[15] J. Sun, H.-Y. Shum and N.-N. Zheng. "Stereo matching using belief propagation", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.25, No.7, pp.787-800, July 2003.

[16] T.M. Breuel, "A comparison of search strategies for geometric branch and bound algorithms", European Conference on Computer Vision (ECCV'02), pp.837-850, 2002.

[17] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Systems, Man, and Cybernetics, vol.6, no.6, pp.420-433, 1976.

[18] B. Luo and E.R. Hancock, "Structural matching using the Em algorithm and singular value decomposition," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.23, pp.1120-1136, 2001.

[19] A. Rangarajan, H. Chui and F.L. Bookstein, "The softassign procrustes matching algorithm", Information Processing in Medical Imaging, James Duncan and Gene Gindi, editors, pp.29-42, Springer, 1997.

[20] J. Pearl, Probabilistic Reasoning in Intelligent Systems – Networks of Plausible Inference, Morgan – Kaufmann 1988.

[21] P.D. Tao, T.Q. Phong, R. Horaud, and L. Quan. "Stability of lagrangian duality for non convex quadratic programming solution methods and applications to computer vision", Mathematical Modelling and Numerical Analysis, vol.31, no.1, pp.57-90, 1997.

[22] X. Bai, H. Yu, and E. Hancock, "Graph matching using embedding and semidefinite Programming", British Machine Vision Conference (BMVC'04), 2004.

[23] J. Kleinberg and E. Tardos. "Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields", IEEE Symposium on Foundations of Computer Science (FOCS'99), pp.14-23, 1999.

[24] M. Ben-Ezra, S. Peleg, and M. Werman, "Real-time motion analysis with linear programming", International Conference on Computer Vision (ICCV'99), pp.703-709, 1999.

[25] D.G. Lowe, "Distinctive image features from scale-invariant keypoints", International Journal of Computer Vision, vol.60, no.2, pp.91-110, 2004.

[26] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts", International Conference on Computer Vision (ICCV'03), p.26, 2003.

[27] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of state calculations by fast computing machines", J. Chem. Phys., Vo.21, No.6, pp.1087-1092, 1953.

[28] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, Linear Programming and Network flows, Wiley Intersicence, 2005.

[29] B.K.P. Horn and B.G. Schunck, Determining optical flow, AI Journal, Vol.17, 1981.

[30] Lucas B.D. Generalized Image Maching by the Method of Differences. PhD Dissertation, Dept. of Computer Science, Carnegie Mellon University.

[31] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, Performance of optical flow techniques, IJCV, Vol.12, pp.43-77, 1994.

[32] Haiying Liu, Chellappa, R., Rosenfeld, A., Accurate dense optical flow estimation using adaptive structure tensors and a parametric model, IEEE Tran. on Image Proc., Vol.12, pp.1170-1180, 2003.

[33] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts", European Conference on Computer Vision (ECCV'02), p.III: 82-96, 2002.

[34] S. Roy and I.J. Cox, "A maximum flow formulation of the N-camera stereo correspondence problem", International Conference on Computer Vision (ICCV'98), pp.429-499, 1998.

[35] C.F. Olson, "Maximum-likelihood image matching", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 24, p. 853-857, 2002.

[36] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts", International Conference on Computer Vision (ICCV'01), pp.II.508-515, 2001.

[37] S.T. Barnard, "Stochastic stereo matching over scale", International Journal of Computer Vision, vol.3, pp.17-32, 1989.

[38] J. Maciel and J.P. Costeira, "A global solution to sparse correspondence problems", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.25, pp.187-199, 2003.

[39] M. Ben-Ezra, S. Peleg, M. Werman, "Real-time motion analysis with linear-programming", International Conference on Computer Vision (ICCV'99), pp.II.703-709, 1999.

[40] M. Ben-Ezra, S. Peleg and M. Werman, "Model based pose estimator using linear programming", European Conference on Computer Vision (ECCV'00), pp.267-281, 2000.

[41] V. Chvatal. Linear Programming, W.H. Freeman and Co., New York, 1983.

[42] M. Kass, A.Wikin, and D. Terzopaulos, "Snakes: active contour models", International Journal of Computer Vision, 1:321-331, 1988.

[43] H. Jiang and M.S. Drew, "Shadow-resistant tracking in video", International Conference on Multimedia and Expo (ICME'03), pp. III 77-80, 2003.

[44] C. Xu and J.L. Prince, "Snake, shapes, and gradient vector flow", IEEE Trans. Image Proc., 7:359-369, 1998.

[45] A.C. Berg, T. L. Berg, J. Malik "Shape matching and object recognition using low distortion correspondence", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005

[46] Kidsroom – An Interactive Narrative Playspace. `http://vismod.media.mit.edu/vismod/demos/kidsroom/kidsroom.html`.

[47] A.P. Pentland, C.R. Wren, F. Sparacino, A.J. Azarbayejani, T.J. Darrell, T.E. Starner, A. Kotani, C.M. Chao, M. Hlavac, K.B. Russell, "Perceptive spaces for performance and entertainment: Untethered interaction using computer vision and audition", Applied Artificial Intelligence, v.11 no.4, pp.267-284, 1997.

[48] L. Emering and B. Herbelin, "Body gesture recognition and action response", Handbook of Virtual Humans, Wiley 2004, pp. 287-302.

[49] VIVID GROUP gesture recognition system. http://www.vividgroup.com.

[50] F. Sparacino, N. Oliver, A. Pentland, "Responsive portraits". The Eighth International Symposium on Electronic Art, Chicago, IL, USA, September 22-27, 1997.

[51] K.M.G. Cheung, S. Baker, T. Kanade, "Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), pp. I:77-84 vol. 1, 2003.

[52] R. Ronfard, C. Schmid, and B. Triggs, "Learning to parse pictures of people", European Conference on Computer Vision (ECCV'02), LNCS 2353, pp.700-714, 2002.

[53] G. Mori, X. Ren, A. Efros, and J. Malik, "Recovering human body configurations: combining segmentation and recognition", IEEE Conference on Computer Vision and Pattern Recognition (ICCV'04), pp.II:326-333, 2004.

[54] D. Ramanan, D. A. Forsyth, and A. Zisserman. "Strike a pose: tracking people by finding stylized poses", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005.

[55] D. M. Gavrila and V. Philomin, "Real-time object detection for smart vehicles", International Conference on Computer Vision (ICCV'99), pp.87-93, Kerkyra, Greece, 1999.

[56] G. Mori and J. Malik, "Estimating human body configurations using shape context matching", European Conference on Computer Vision (ECCV'02), LNCS 2352, pp. 666-680, 2002.

[57] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts", IEEE Trans. Pattern Analysis and Machine Intelligence, vol.24, pp.509-522, 2002.

[58] H. Jiang, Z.N. Li and M.S. Drew, "Linear programming for matching in human body gesture recognition", Analysis and Modeling of Faces and Gestures (AMFG'05), LNCS 3723, pp. 392-406, 2005.

[59] H. Jiang, Z.N. Li and M.S. Drew, "Human posture recognition with convex programming", IEEE International Conference on Multimedia and Expo (ICME'05), 2005.

[60] H. Jiang, M. S. Drew, and Z.-N. Li, "Linear programming matching and appearance-adaptive object tracking", Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR'05), LNCS 3757, pp. 203-219, 2005.

[61] A. F. Bobick and S. S. Intille. "Large occlusion stereo", International Journal of Computer Vision, 33(3):181-200, 1999.

[62] Ning Xu, Ravi Bansal and Narendra Ahuja, "Object segmentation using graph cuts based active contours", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), vol. 2, pp. 46-53, 2003.

[63] Ning Xu and Narendra Ahuja, "Object contour tracking using graph cuts based active contours", IEEE International Conference on Image Processing (ICIP'02), vol 3, pp. 277-280, 2002.

[64] Kang Li, Xiaodong Wu, Danny Z. Chen, Milan Sonka, "Globally Optimal Segmentation of Interacting Surfaces with Geometric Constraints", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), vol.1, pp.394-399, 2004.

[65] O. Faugeras, R. Keriven, "Variational principles, surface evolution, PDEs, level set methods, and the stereo problem", IEEE Trans. on Image Proc., Vol. 7, no. 3, 1998. pp.336-344.

[66] J. Maciel and J Costeira, "A global solution to sparse correspondence problems", IEEE Trans. on Pattern Analysis and Machine Intelligence, 25(2):187 199, 2003

[67] M. J. Black and A. D. Jepson. "Eigentracking: robust matching and tracking of articulated objects using a view-based representation", European Conference on Computer Vision (ECCV'96), pp.329-342, 1996.

[68] L.P. Morency, A. Rahimi, and T. Darrell, "Adaptive view-based appearance models", IEEE Conference on Computer Vision and Pattern Recognition, I:803-810, 2003.

[69] J. Chuzhoy and S. Naor. "The hardness of metric labeling", IEEE Symposium on Foundations of Computer Science (FOCS'04), pp.108-114, 2004.

[70] P. Blomgren, T.F. Chan, "Color TV: total variation methods for restoration of vector-valued images", IEEE Trans. Image Proc., vol.7, no.3, pp. 304-309.

[71] T.F. Chan, Chiu-Kwong Wong, "Total variation blind deconvolution", IEEE Trans. Image Proc., vol.7, no.3, pp. 370-375.

[72] S.L. Lauritzen, D.J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems", Journal of the Royal Statistical Society, Series B (Methodological), 50(2):157-224.

[73] A.C. Berg, T.L. Berg, J. Malik, "Shape matching and object recognition using low distortion correspondence", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005.

[74] P.F. Felzenszwalb, D.P. Huttenlocher, "Efficient matching of pictorial structures", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), pp.II:66-73 vol.2, 2000.

[75] A. Mahaloanobis, B.V.K. Vijaya Kumar et cl., "Unconstrained Correlation Filters", Appl. Opt., 33:3751-3759, 1994.

[76] D. Comaniciu, V. Ramesh, and P. Meer. "Real-time tracking of non-rigid objects using mean shift", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), II:142-149, 2000.

[77] Xiaofeng Ren, Alexander C. Berg, Jitendra Malik, "Recovering Human Body Configurations using Pairwise Constraints Between Parts", International Conference on Computer Vision (ICCV'05), Beijing, pp. 824-831, 2005.

[78] G. Mori, Y. Wang, H. Jiang, M.S. Drew, Ze-Nian Li, "Unsupervised discovery of action classes", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, to appear.

[79] H. Jiang, M.S. Drew and Z.N. Li, "Successive convex matching for action detection", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006, to appear.

[80] H. Jiang, Z.N. Li and M.S. Drew, "Detecting human action in active video", International Conference on Multimedia and Expo (ICME'06), 2006, to appear.

[81] H. Jiang, Z.N. Li and M.S. Drew, "Successive Convex Quadratic Programming for Object Localization", International Conference on Pattern Recognition (ICPR'06), 2006, to appear.

[82] R.Y. Tsai,"An efficient and accurate camera calibration technique for 3D machine vision", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'86), pp.364–374, 1986.

[83] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach", International Conference on Pattern Recognition (ICPR'04), 2004.

[84] J. O'Rourke, Computational Geometry in C, 2nd edition, Cambridge Univeristy Press, 1998.

[85] B. Lucas and T. Kanada, "An iterative image registration technique with an application to stereo vision", International Joint Conference on Artificial Intelligence (IJCAI'81), pp.674-679, 1981.

[86] O. Nestares, D. Fleet, and D. Heeger, "Likelihood functions and confidence bounds for total-least-squares estimation", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), vol.1, pp. 523-530, 2000.

[87] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift", IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), pp.142-151, 2000.

[88] D. Murray and A. Busu, "Motion tracking with an active camera", IEEE Trans. on Pattern Analysis and Machine Intelligence, 16(5):449-459, 1994.

[89] P.H.S. Torr and A. Criminisi, "Dense Stereo using Pivoted Dynamic Programming", Image and Vision Computing, 22(10), pp.795-806, 2004.

[90] Wasserstrom, E. "Numerical solutions by the continuation method", SIAM Review, vol.15, pp.89-119, 1973.