# Spatial, Temporal and Spatio-temporal Correspondence for Computer Vision Problems

Feng Zhou

CMU-RI-TR-14-18

September 24

The Robotics Institute

Carnegie Mellon University

Pittsburgh, PA 15213

**Thesis Committee:**

Fernando De la Torre (Chair)

Martial Hebert

Jeff Schneider

Anand Rangarajan (University of Florida)

*Submitted in partial fulfillment of the*
*requirements for the degree of*
*Doctor of Philosophy in Robotics*

**Keywords:** Correspondence Problem, Temporal Alignment, Graph Matching, Dynamic Time Warping, Quadratic Assignment Problem, Trajectory Matching

## Abstract

Many computer vision problems, such as object classification, motion estimation or shape registration rely on solving the correspondence problem. Existing algorithms to solve spatial or temporal correspondence problems are usually NP-hard, difficult to approximate, lack flexible models and mechanism for feature weighting. This proposal addresses the correspondence problem in computer vision, and proposes two new spatio-temporal correspondence problems and three algorithms to solve spatial, temporal and spatio-temporal matching between video and other sources. The main contributions of the thesis are:

(1) Factorial graph matching (FGM). FGM extends existing work on graph matching (GM) by finding an exact factorization of the affinity matrix. Four are the benefits that follow from this factorization: (a) There is no need to compute the costly (in space and time) pairwise affinity matrix; (b) It provides a unified framework that reveals commonalities and differences between GM methods. Moreover, the factorization provides a clean connection with other matching algorithms such as iterative closest point; (c) The factorization allows the use of a path-following optimization algorithm, that leads to improved optimization strategies and matching performance; (d) Given the factorization, it becomes straight-forward to incorporate geometric transformations (rigid and non-rigid) to the GM problem.

(2) Canonical time warping (CTW). CTW is a technique to temporally align multiple multi-dimensional and multi-modal time series. CTW extends DTW by incorporating a feature weighting layer to adapt different modalities, allowing a more flexible warping as combination of monotonic functions, and has linear complexity (unlike DTW that has quadratic). We applied CTW to align human motion captured with different sensors (e.g., audio, video, accelerometers).

(3) Spatio-temporal matching (STM). Given a video and a 3D motion capture model, STM finds the correspondence between subsets of video trajectories and the motion capture model. STM is efficiently and robustly solved using linear programming. We illustrate the performance of STM on the problem of human detection in video, and show how STM achieves state-of-the-art performance.

# Acknowledgments

My foremost thanks go to my advisor, Prof. De la Torre, for his adequate support and guidance to enable the thesis to be completed. I'm thankful for the freedom he gave me to pursue a variety of topics. I've learned a tremendous amount from him during my journey, and I'm eternally grateful and in debt for his guidance, support, and patience with me along the way.

I would also like to thank Martial Hebert, Jeff Schneider, Anand Rangarajan for their interests in my work and for serving on my committee. I appreciate the time they set aside for me and am thankful for the invaluable discussions and comments regarding this work.

Over the years I've been fortunate to have being working with my colleagues from Human Sensing Lab at CMU: Dong Huang, Vincent Chu, Xuehan Xiong, Ricardo Silveira Cabral, Jayakorn Vongkulbhisal, Ada Zhang, Ekaterina Spriggs, Zehua Huang, Shitong Yao, Francisco Vicente; and to my former colleagues: Zengyin Zhang, Javier Hernandez, Tomas Simon, Yunfeng Zhu, Xuelei Hu, Caikou Chen, Ying Chen. They made the Human Sensing Lab a fun and exciting place to work. Thanks for providing help in research related discussions during numerous coffee, lunch, and dinner breaks.

This work was funded by the Quality of Life Technology (QoLT) center. Thanks to all QoLT members for interesting discussions and highly appreciated feedback during research meetings and mutual visits.

Last but not least, I want to thank my parents and my wife for years of support and encouragement.

# Contents

# List of Figures

# List of Tables

# Notation

| | |
|---|---|
| $\mathbf{X}$ | Bold capital letters denote a matrix. |
| $\mathbf{x}$ | Bold lower-case letters denote a vector. |
| $x$ | Non-bold letters denotes a scalar. |
| $x_{ij}$ or $[\mathbf{X}]_{ij}$ | Scalar in the $i^{th}$ row and $j^{th}$ column of the matrix $\mathbf{X}$. |
| $\mathbf{1}_{m\times n} \in \mathbb{R}^{m\times n}$ | Matrix containing value one. |
| $\mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$ | Matrix containing value zero. |
| $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ | Identity matrix. |
| $\lvert\mathbf{X}\rvert$ | Determinant of the square matrix $\mathbf{X}$. |
| $\lVert\mathbf{x}\rVert_p = \sqrt[p]{\sum_i \lvert x_i\rvert^p}$ | $p$-norm of the vector $\mathbf{x}$. |
| $\lVert\mathbf{X}\rVert_F = \sqrt{\mathrm{tr}(\mathbf{X}^T\mathbf{X})}$ | Frobenious norm of the matrix $\mathbf{X}$. |
| $\mathrm{vec}(\mathbf{X})$ | Vectorization of the matrix $\mathbf{X}$. |
| $\mathrm{diag}(\mathbf{x})$ | A diagonal matrix whose diagonal elements are $\mathbf{x}$. |
| $\mathbf{X} \circ \mathbf{Y}$ | Hadamard products of matrices. |
| $\mathbf{X} \otimes \mathbf{Y}$ | Kronecker products of matrices. |
| $\{i : j\}$ | A list of integers, $\{i, i+1, \cdots, j-1, j\}$. |
| $[\mathbf{X}_1; \cdots; \mathbf{X}_n]$ | Vertical concatenation of matrices. |
| $[\searrow_i \mathbf{X}_i]$ | Diagonal concatenation of matrices. |
| $\mathbf{X}^\dagger$ | Moore-Penrose pseudo-inverse of matrix $\mathbf{X}$. |
| $\mathrm{eig}_d(\mathbf{A}, \mathbf{B})$ | Top $d$ eigenvectors $\mathbf{V}$ that solve the generalized eigenvalue problem $\mathbf{AV} = \mathbf{BV}\Lambda$, where $\Lambda$ denote the eigenvalues. |

# Chapter 1

# Introduction

In the last decade, the amount of data captured by various visual sensors (*e.g.*, hand-held camera, smart-phone, Kinect, Google Glass, GoPro) has been growing exponentially. It has thus become increasingly important to develop effective and efficient systems that can understand massive amounts of images, videos and depth data. To detect and recognize spatial objects or temporal events from these data, most existing methods build upon the popular bag-of-words (BoW) representation [109, 139, 177]. Although the technique is relatively simple and easy to understand, BoW achieves great success, for example, in classifying natural images and videos, such as feature films [109], broadcast sports [158], and YouTube [121]. However, BoW-type methods lack the fundamental mechanisms to enforce spatial or temporal consistency across images and videos. When classifying spatial objects or temporal events with complex internal structures, it is not always possible to construct BoW-type vectors that capture important discriminative information between object classes or human activities.

In recent research, these difficulties led to the use of richer representations, which encode object features and the structural relationships between them. The middle column in Fig. 1.1 summarizes several popular structural representations of images and videos. For instance, tree-like representations have been widely used to describe the geometrical configuration of human bodies [68, 208] and faces [228] due to the efficiency in inference. For objects with more complicated structures, graphs [44, 117] are more commonly used to encode the spatial arrangement of features. Because of the monotonicity in time, temporal video and motion capture sequences

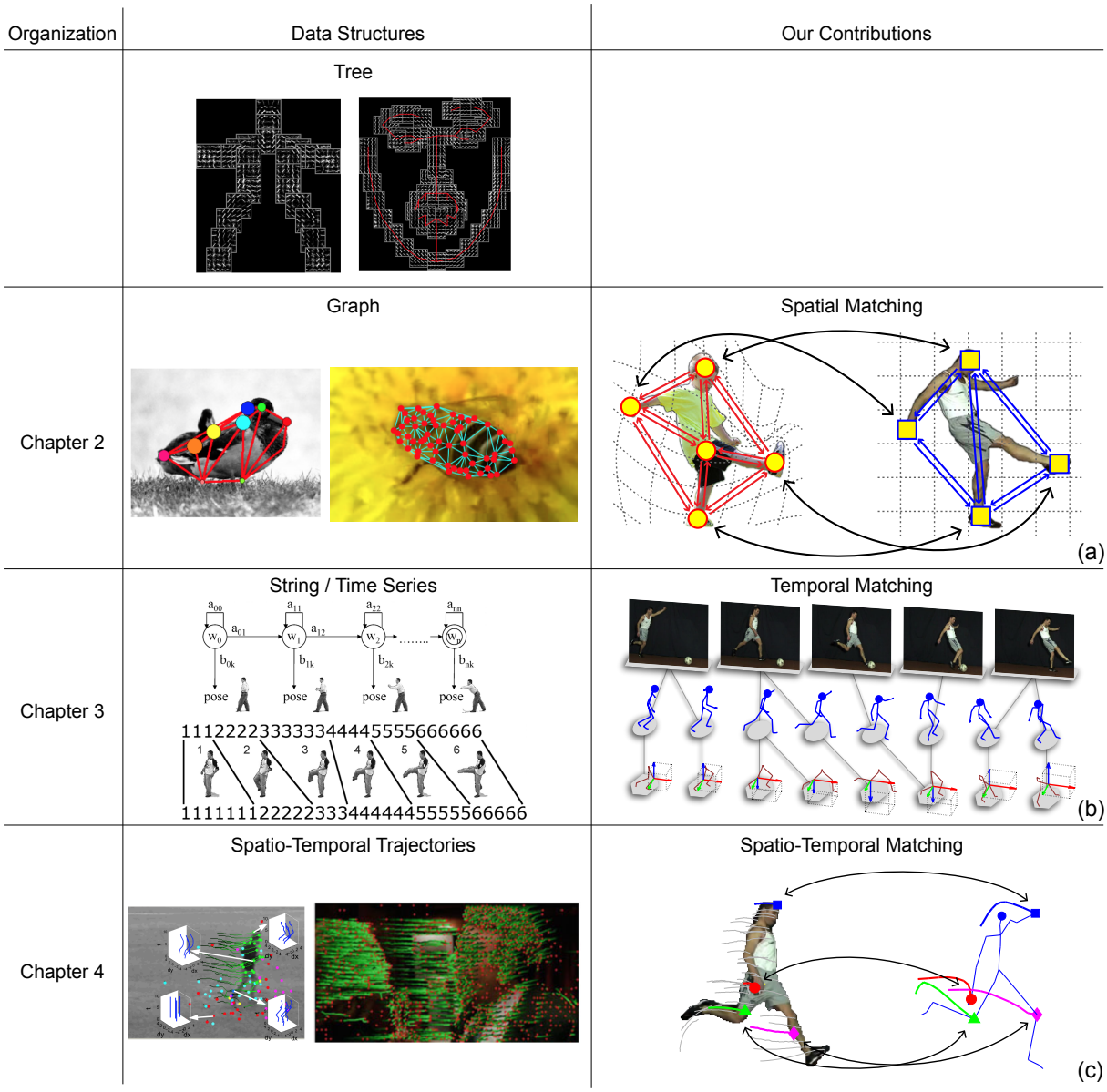


Figure 1.1: Thesis outlines and contributions. Given the various structural representations (*e.g.*, tree, graph, string and spatio-temporal trajectories) of images and videos, this thesis addresses three correspondence problems: (a) Spatial matching between features of images; (b) Temporal alignment of multi-modal sequences; and (c) Spatio-temporal matching between feature trajectories extracted from 2-D video and 3-D motion capture sequences. Images credited to [5, 44, 117, 130, 146, 198, 208].

are usually encoded as strings or multi-dimensional time series [5]. But interpreting videos that show complex human activities requires reasoning about the spatio-temporal arrangement of body parts tracked over video [130, 198].

Once a structural representation has been constructed, a natural question arising is the establishment of the correspondence between two instances which optimally aligns their structures. In the past decade, various correspondence models have been proposed to address a wide range of computer vision problems, including shape matching [18, 129], object categorization [61, 114], feature tracking [95], symmetry analysis [85], video synchronization [200], action detection [93] and recognition [29, 75]. It has also been shown that current BoW-typed models can be greatly improved [38, 61, 114] when knowing the correspondence between images. In addition to being useful for computer vision, establishing correspondence is a crucial step in many other pattern recognition problems, such as kernelized sorting [147], protein alignment [215] to speech recognition [148].

## 1.1 Challenges

Although extensive studies have been done, solving correspondence problems in computer vision is still challenging for several reasons.

**Optimizing combinatorial problems.** Correspondence problems involve optimization over discrete variables with combinatorial constraints. In general, correspondence problems are often NP-hard and exact solutions are infeasible. Therefore, the main body of the past research has focused on devising more accurate and faster algorithms to approximate these problems. Nevertheless, approximating correspondences by relaxing the combinatorial constraints is still challenging in many cases when objective is non-convex or discontinuous.

**Handling geometrical constraints.** Many correspondence problems in computer vision naturally require global constraints among nodes. For instance, given two sets of coplanar points in two images, the matching between points should be constrained by an affine transformation. Similarly, when matching human bodies between 2D video and 3D motion capture data, the transformation needs to be constrained as perspective projection. However, solving correspon-

dence problems under global geometrical constraints result to mixed-integer programming and existing algorithms are prone to local optima.

**Selecting important features.** Correspondence problems are complicated in unconstrained cases due to the large variance in subject's physical characteristics, motion style and camera configuration. In addition, the problems can be further complicated when matching between multi-modal data (*e.g.*, video, image, motion capture, accelerometer, depth data). However, existing correspondence methods lack a feature selection mechanism to select important features that accommodate for subject variability and take into account the difference in the dimensionality of the signals.

## 1.2 Our contributions

In this thesis, we focus on three important correspondence problems (the right column in Fig. 1.1) in computer vision: matching spatial features between images, matching temporal frames between multi-modal sequences (*e.g.*, video, mocap), and matching spatio-temporal feature trajectories between 2-D video and 3-D mocap sequence. We describe these problems in detail below.

**Spatial matching.** Finding correspondences between spatial features is essential for detecting and recognizing objects from images. When the spatial configuration of an object is represented as a graph (Fig. 1.1a), the problem of spatial matching can be formalized as graph matching [48]. However, graph matching is a classical combinatorial problem [124], and the computational complexity of existing methods limits the permissible size of input graphs in practice. This thesis presents factorized graph matching (FGM), a novel framework for interpreting and optimizing graph matching problems. This work reveals a general factorization to decouple any type of graph matching problem into small-scale components. Based on this factorization, we develop an efficient and accurate technique that improves state-of-the-art algorithms in graph matching.

**Temporal matching.** Temporal matching of temporal signals is a fundamental step in many applications such as activity recognition [97] and human motion synthesis [88]. Major chal-

lenges for aligning human motion include: (1) introducing a feature selection to compensate for the differences in subjects' physical characteristics, motion styles and speed of actions; and (2) allowing alignment between different features (*e.g.*, video and mocap). For instance, how can we solve for the temporal correspondence between the frames of a video, the samples of motion capture data and the accelerometer signal from different people kicking a ball (Fig. 1.1b)? This thesis present canonical time warping (CTW), which combines canonical correlation analysis with dynamic time warping to simultaneously align two signals in time while performing feature selection. Using CTW, we are able to address the difficulty of aligning human actions from video, motion capture data, or video and motion capture data under camera view or style changes. We further show how to generalize CTW to align multiple sequences and develop a more efficient algorithm that reduces the quadratic complexity incurred by conventional time warping method to a linear complexity.

**Spatio-temporal matching.** Detecting and tracking humans in videos have been long-standing problems in computer vision. Most successful approaches (*e.g.*, deformable parts models [69, 208]) rely heavily on discriminative models to build appearance detectors for body joints and generative models to constrain possible body configurations. While these 2D models have been successfully applied to images (and with less success to videos), a major challenge in generalizing these models to handle different camera views. In order to achieve view-invariance, these 2D models typically require a large amount of training data across views, which is difficult to gather and time-consuming to label. Unlike existing 2D models, this thesis formulates the problem of human detection in videos as spatio-temporal matching (STM) between a 3D motion capture model and trajectories in videos. Our algorithm estimates the camera view and selects a subset of tracked trajectories that matches the motion of 3D model (see Fig. 1.1c for an example). The STM is efficiently solved with linear programming, and it is robust to tracking mismatches, occlusions and outliers. To the best of our knowledge, we are the first to solve the correspondence between video and 3D motion capture data for human pose detection.

## 1.3 Organization

The remainder of this dissertation is structured as follows. In Chapter 2, we propose the method of factorized graph matching and quantitatively evaluate its performance in comparison with state-of-the-art graph matching methods. In Chapter 3, we propose and quantitatively evaluate the method of canonical time warping for aligning multi-modal sequences. In Chapter 4, we propose a spatial-temporal matching technique for human detection in videos. In Chapter 5 and Chapter 6, we summarize our contributions and discuss future work, respectively.

# Chapter 2

# Spatial Matching

## 2.1 Introduction

Graph matching (GM) plays a central role in many computer science problems, ranging from shape matching in 2-D [18] and 3-D [129], object categorization [61, 114], feature tracking [95], symmetry analysis [85], action recognition [29, 75], kernelized sorting [147] to protein alignment [215]. In computer vision, GM has been primarily used to find correspondences between two sets of features extracted from images. Unlike conventional matching methods such as RANSAC [71, 182] and iterative closest point (ICP) [21, 219], GM incorporates pairwise node interactions which are important features when matching structural objects (*e.g.*, human bodies). Fig. 2.1 illustrates an example of matching two graphs, where the nodes are image locations returned by a body part detector. Matching the graphs using only similarity between nodes (*i.e.*, features) might lead to undesirable results because some features of nodes (*e.g.*, hands, feet) are likely to be similar. GM adds pairwise information between nodes (*e.g.*, length of the limbs, orientation of edges) that constraints the problem and typically better correspondences are found.

Although extensive research has been done for decades, there are still two main challenges in solving GM. (1) Mathematically, GM is formulated as a quadratic assignment problem (QAP) [124]. Unlike the linear assignment problem, which can be efficiently solved with the Hungarian algorithm [34], the QAP is known to be NP-hard. Therefore, the main body of research in GM has focused on devising more accurate algorithms to solve it approximately. Nevertheless, ap-

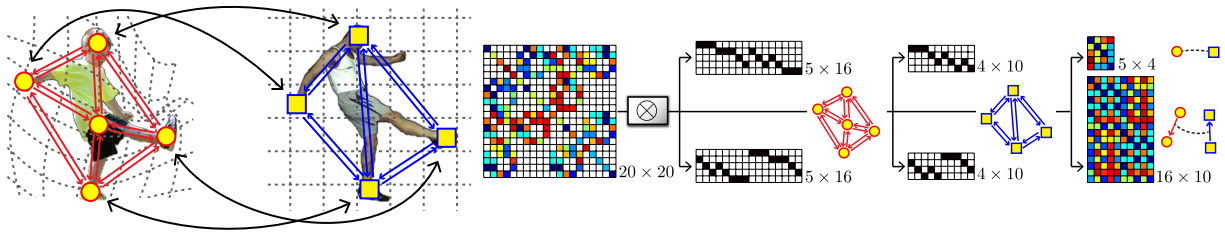


Figure 2.1: Matching two human bodies with $5$ and $4$ features using FGM. FGM simultaneously estimates the correspondence and a smooth non-rigid transformation between shapes. FGM is able to factorize the $20 \times 20$ pairwise affinity matrix as a Kronecker product of six smaller matrices. The first two groups of matrices of size $5 \times 16$ and $4 \times 10$ encode the structure of each of the graphs. The last two matrices encode the affinities for nodes ($5 \times 4$) and edges ($16 \times 10$).

proximating GM by relaxing the combinatorial constraints is still a challenging problem. This is mainly because the objective function is in general non-convex and thus existing methods are prone to local optima. (2) Many matching problems in computer vision naturally require global constraints among nodes. For instance, given two sets of coplanar points in two images, the matching between points under orthographic projection should be constrained by an affine transformation. Similarly, when matching the deformations of non-rigid objects between two consecutive images, that deformation is typically smooth in space and time. Existing GM algorithms do not constrain the nodes of both graphs to a given geometric transformation (*e.g.*, similarity, affine or non-rigid).

In order to address these issues, this chapter presents factorized graph matching (FGM), a novel framework for optimizing and constraining GM problems. The key idea behind FGM is a closed-form factorization of the pairwise affinity matrix that decouples the local graph structure of the nodes and edge similarities. This factorization is general and can be applied to both undirected and directed graphs. For instance, consider the matching of the two human bodies shown in Fig. 2.1. The body configurations are represented by two directed graphs with $5$ and $4$ features, and $16$ and $10$ edges respectively. Conventional GM algorithms need to construct a large pairwise affinity matrix (20-by-20 in this case). Whereas FGM only requires six small matrices to be computed. In our example (Fig. 2.1), the first two binary matrices are of dimension 5-by-16 and the second two of dimensions 4-by-10 and describe the local structure of the first

and second graph. The last two matrices are of dimensions $5$-by-$4$ and $16$-by-$10$, and they encode the similarity between nodes and edges respectively.

Using this factorization has four benefits: First, there is no need to compute the expensive (in space and time) affinity matrix, which scales quartically with the number of features. Second, it provides a unified view of many GM methods, which allows to understand the commonalities and differences between them. It also connects GM methods with the classical Iterative Closest Point (ICP) algorithms, and provides a pairwise generalization of ICP. Third, it allows the use of a path-following algorithm, that leads to improved optimization strategies and matching performance. Fourth, it is easy to add global geometric constraints because we have decoupled the local structure for the nodes in each graph. We illustrate the benefits of FGM in synthetic and real matching experiments on several benchmark databases.

The structure of the chapter is organized as follows. Section 2.2 and section 2.3 review related work in GM and ICP respectively. Section 2.4 introduces the factorization of the pairwise affinity matrix. Section 2.5 discusses a unified taxonomy among various GM algorithm as well as the connection between ICP and Markov random field (MRF) using the factorization. Section 2.6 derives a path-following algorithm from the factorization for optimizing GM. Section 2.7 extends GM problems by introducing global geometrical constraints between graphs. Section 2.8 compares FGM with state-of-the-art methods on several benchmark datasets. Preliminary versions of this chapter were published in [222, 224].

## 2.2 Previous work on graph matching

### 2.2.1 Definition of GM

We denote a graph with $n$ nodes and $m$ directed edges as a $4$-tuple $\mathcal{G} = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}, \mathbf{H}\}$. The features for nodes and edges are specified by $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{d_p \times n}$ and $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m] \in \mathbb{R}^{d_q \times m}$ respectively, where $d_p$ and $d_q$ are the dimensionality of the features. For instance, $\mathbf{p}_i$ could be a $128$-D SIFT histogram extracted from the image patch around the $i^{th}$ node and $\mathbf{q}_c$ could be a $2$-D vector that encodes the length and orientation of the $c^{th}$ edge. The topology of the graph is encoded by two node-edge incidence matrices $\mathbf{G}, \mathbf{H} \in \{0, 1\}^{n \times m}$, where $g_{ic} = h_{jc} = 1$ if the $c^{th}$

edge starts from the $i^{th}$ node and ends at the $j^{th}$ node. Fig. 2.2a illustrates two synthetic graphs, whose edge connection between nodes is encoded by the binary matrices shown in Fig. 2.2b-c. In our previous work [222], a simpler representation of graph was adopted and valid only for undirected graphs. This representation is more general and applicable for both undirected and directed graphs. Directed graphs typically occur when the edge features are asymmetrical such as the angle between an edge and the horizontal line.

Given a pair of graphs, $\mathcal{G}_1 = \{\mathbf{P}_1, \mathbf{Q}_1, \mathbf{G}_1, \mathbf{H}_1\}$ and $\mathcal{G}_2 = \{\mathbf{P}_2, \mathbf{Q}_2, \mathbf{G}_2, \mathbf{H}_2\}$, we compute two affinity matrices, $\mathbf{K}_p \in \mathbb{R}^{n_1 \times n_2}$ and $\mathbf{K}_q \in \mathbb{R}^{m_1 \times m_2}$, to measure the similarity of each node and edge pair respectively. More specifically, $\kappa^p_{i_1 i_2} = \phi_p(\mathbf{p}^1_{i_1}, \mathbf{p}^2_{i_2})$ measures the similarity between the $i_1^{th}$ node of $\mathcal{G}_1$ and the $i_2^{th}$ node of $\mathcal{G}_2$, and $\kappa^q_{c_1 c_2} = \phi_q(\mathbf{q}^1_{c_1}, \mathbf{q}^2_{c_2})$ measures the similarity between the $c_1^{th}$ edge of $\mathcal{G}_1$ and the $c_2^{th}$ edge of $\mathcal{G}_2$. For instance, Fig. 2.2d illustrates an example pair of $\mathbf{K}_p$ and $\mathbf{K}_q$ for the two synthetic graphs.

Given two graphs and the associated affinity matrices, the problem of GM consists in finding the optimal correspondence $\mathbf{X}$ between nodes, such that the sum of the node and edge compatibility is maximized:

$$J_{gm}(\mathbf{X}) = \sum_{i_1 i_2} x_{i_1 i_2} \kappa^p_{i_1 i_2} + \sum_{\substack{i_1 \neq i_2, j_1 \neq j_2 \\ g^1_{i_1 c_1} h^1_{j_1 c_1} = 1 \\ g^2_{i_2 c_2} h^2_{j_2 c_2} = 1}} x_{i_1 i_2} x_{j_1 j_2} \kappa^q_{c_1 c_2}, \tag{2.1}$$

where $\mathbf{X} \in \Pi$ is constrained to be a one-to-one mapping, *i.e.*, $\Pi$ is the set of partial permutation matrices:

$$\Pi = \{\mathbf{X} | \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{X}^T \mathbf{1}_{n_1} = \mathbf{1}_{n_2}\}. \tag{2.2}$$

The inequality in the above definition is used for the case when the graphs are of different sizes. Without loss of generality, we assume $n_1 \geq n_2$ throughout the rest of the chapter. For instance, the matrix $\mathbf{X}$ shown in the top row of Fig. 2.2e defines the node correspondence shown in Fig. 2.2a.

To be concise in notation, we encode the node and edge affinities in a symmetrical matrix

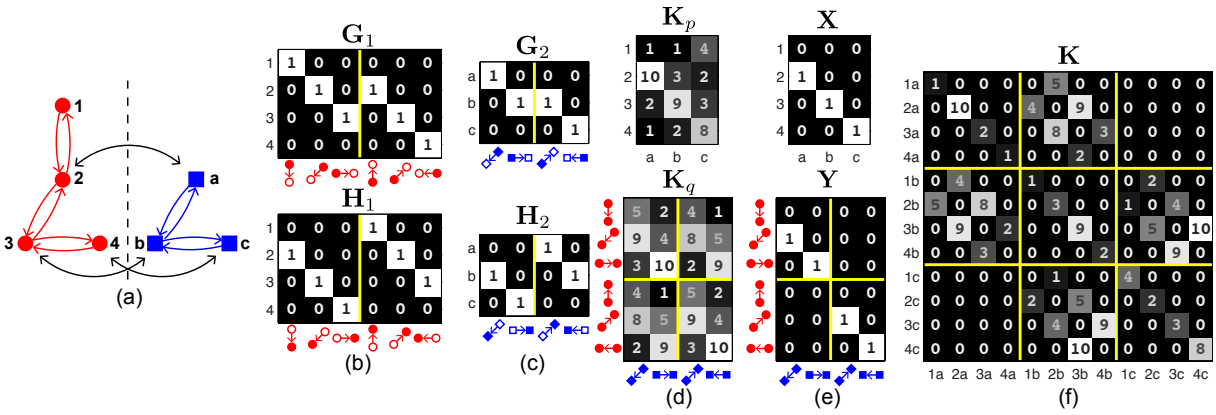


Figure 2.2: An example GM problem. (a) Two synthetic graphs. (b) The $1^{st}$ graph's incidence matrices $\mathbf{G}_1$ and $\mathbf{H}_1$, where the non-zero elements in each column of $\mathbf{G}_1$ and $\mathbf{H}_1$ indicate the starting and ending nodes in the corresponding directed edge, respectively. (c) The $2^{nd}$ graph's incidence matrices $\mathbf{G}_2$ and $\mathbf{H}_2$. (d) The node affinity matrix $\mathbf{K}_p$ and the edge affinity matrix $\mathbf{K}_q$. (e) The node correspondence matrix $\mathbf{X}$ and the edge correspondence matrix $\mathbf{Y}$. (f) The global affinity matrix $\mathbf{K}$.

$\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, whose elements are computed as follows:

$$\kappa_{i_1 i_2, j_1 j_2} = \begin{cases} \kappa^p_{i_1 i_2}, & \text{if} \quad i_1 = j_1 \text{ and } i_2 = j_2, \\ \kappa^q_{c_1 c_2}, & \text{if} \quad i_1 \neq j_1 \text{ and } i_2 \neq j_2 \text{ and} \\ & \quad g^1_{i_1 c_1} h^1_{j_1 c_1} g^2_{i_2 c_2} h^2_{j_2 c_2} = 1, \\ 0, & \text{otherwise}, \end{cases}$$

where the diagonal and off-diagonal elements encode the similarity between nodes and edges respectively.

Using $\mathbf{K}$, GM can be concisely formulated as the following quadratic assignment problem (QAP) [124]:

$$\max_{\mathbf{X} \in \Pi} \quad J_{gm}(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^T \mathbf{K}\, \mathrm{vec}(\mathbf{X}). \tag{2.3}$$

### 2.2.2 Advances on GM

As a generic problem for matching structural data, GM has been studied for decades in computer science and mathematics [48]. Early works [49, 144, 187] on GM often treated the problem as

a special case of (sub)graph isomorphism, where the graphs are binary and an exact matching between nodes and edges is of interest. Nevertheless, the stringent constraints imposed by exact matching are too rigid for real applications in computer vision. Modern works focus on finding an inexact matching between graphs with weighted attributes on nodes and edges. Due to the combinatorial nature, however, globally optimizing GM is NP-hard. A relaxation of the permutation constraints (Eq. 2.2) is necessary to find an approximation to the problem. Based on the approximation strategy, previous work can be broadly categorized in three groups: spectral relaxation, semidefinite-programming (SDP) relaxation and doubly-stochastic relaxation. See Fig. 2.1 for a summary of previous relaxations of GM.

The first group of methods approximates the permutation matrix with an orthogonal one[1], *i.e.*, $\mathbf{X}^T\mathbf{X} = \mathbf{I}$. Under the orthogonal constraint, optimizing GM can be solved in closed-form as an eigen-value problem [35, 167, 170, 188]. However, these methods can only work for the Koopmans-Beckmann's QAP [105], a special case of QAP (See Section 2.5 for more details). In order to handle more complex problems in computer vision, Leordeanu and Hebert [112] proposed to approximate Eq. 2.3 by relaxing $\mathbf{X}$ to be of unit length, *i.e.*, $\|\operatorname{vec}(\mathbf{X})\|_2^2 = 1$. In this case, the optimal $\mathbf{X}$ can be efficiently computed as the leading eigen-vector of $\mathbf{K}$. Cour *et al.* [51] incorporated an additional affine constraints to solve a more general spectral problem, hence finding better approximations to the original problem.

As a general tool for approximating combinatorial problems, SDP was also used to approximate GM. In [165, 183], the authors reformulated the objective of GM by introducing a new variable $\mathbf{Y} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ subject to $\mathbf{Y} = \operatorname{vec}(\mathbf{X}) \operatorname{vec}(\mathbf{X})^T$. SDP approximates GM by relaxing the non-convex constraint on $\mathbf{Y}$ as a convex semi-definite one, $\mathbf{Y} - \operatorname{vec}(\mathbf{X}) \operatorname{vec}(\mathbf{X})^T \succeq 0$. After computing $\mathbf{Y}$ using SDP, the correspondence $\mathbf{X}$ can be approximated by a randomized algorithm [183] or a winner-take-all strategy [165]. The main advantage of using SDP is its theoretical guarantees [76] to find a polynomial time $0.879$ approximation to many NP-hard problems. However, in practice it is often too expensive to use SDP approaches because the new variable $\mathbf{Y}$ squares the problem size.

The majority of methods relax $\mathbf{X} \in \mathcal{D}$ to be a doubly stochastic matrix, the convex hull of

[1]Mathematically, all the columns of an orthogonal matrix are lying on the Stiefel manifolds [154].

the permutation matrices:

$$\mathcal{D} = \{\mathbf{X}|\mathbf{X} \in [0,1]^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} \leq \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} = \mathbf{1}_{n_2}\}. \tag{2.4}$$

Under this constraint, optimizing GM can be treated as a non-convex quadratic programming problem and various strategies have been proposed to find a local optima. Almohamad and Duffuaa [9] approximated the quadratic cost using linear programming. Zaslavskiy *et al.* [214] employed a path-following strategy to approach the non-convex quadratic programming problem. Recently, Liu *et al.* [123] extended the path-following algorithm for matching asymmetrical adjacency matrices in more general problems. However, these works are only applicable to Koopmans-Beckmann's QAP and it is unclear how to apply it to the more general Lawler's QAP problem [110]. Gold and Rangarajan [77] proposed the graduated assignment algorithm to iteratively solve a series of linear approximations of the cost function using Taylor expansions. Its convergence has been recently studied and improved in [180] with a soft constrained mechanism. Van Wyk and Van Wyk [191] proposed to iteratively project the approximate correspondence matrix onto the convex domain of the desired integer constraints. Leordeanu *et al.* [115] proposed an integer projection algorithm to optimize the objective function in an integer domain. Torresani *et al.* [184] designed a complex objective function which can be efficiently optimized by dual decomposition. In addition to the optimization-based work, probabilistic frameworks were shown to be useful for interpreting and solving GM problems. Egozi *et al.*. [62] presented a probabilistic interpretation of the spectral matching algorithm [112], which is a maximum-likelihood estimate of the assignment probabilities. Zass and Shashua [216] proposed a convex relative-entropy error that emerges from a probabilistic interpretation of the GM problem. Inspired by the PageRank algorithm [141], Cho *et al.*. [43] introduced a random-walk algorithm to approximate GM problem. More recently, Suh *et al.* [178] proposed to use sequential Monte Carlo sampling to improve the robustness of GM methods.

In addition to the efforts on devising better approximations for Eq. 2.3, there are three advances in GM methods leading to improved results: learning the pair-wise affinity matrix [37, 116], incorporate high-order features [41, 60, 216], and progressive mechanism [42]. First, it has been shown that a better $\mathbf{K}$ for GM can be learned in an unsupervised [116] or a supervised [37] leading to improved results. Second, it has also been noticed that the matrix $\mathbf{K}$ encoding the pair-

| | $\mathrm{tr}(\mathbf{X}^T\mathbf{A}_1\mathbf{X}\mathbf{A}_2) + \mathrm{tr}(\mathbf{K}_p^T\mathbf{X})$ (Koopmans-Beckmann's QAP) | $\mathrm{vec}(\mathbf{X})^T\mathbf{K}\,\mathrm{vec}(\mathbf{X})$ (Lawler's QAP) |
|---|---|---|
| Spectral Relaxation | Umeyama [188]<br>Scott & Longuet-Higgins [167]<br>Shapiro & Brady [170]<br>Caelli & Kosinov [35] | Leordeanu & Hebert [112]<br>Cour *et al.* [51] |
| SDP Relaxation | | Torr [183]<br>Schellewald & Schnörr [165] |
| Doubly-stochastic Relaxation | Almohamad & Duffuaa [9]<br>Zaslavskiy *et al.* [214]<br>Liu *et al.* [123] | Gold & Rangarajan [77]<br>Tian *et al.*. [180]<br>Van Wyk & Van Wyk [191]<br>Leordeanu *et al.* [115]<br>Torresani *et al.* [184]<br>Egozi *et al.*. [62]<br>Zass & Shashua [216]<br>Cho *et al.* [43]<br>Suh *et al.* [178]<br>Ours |

Table 2.1: Different relaxations on the constraints for GM.

wise geometry is susceptible to scale and rotation differences between sets of points. In order to make GM invariant to rigid deformations, [41, 60, 216] extended the pairwise matrix $\mathbf{K}$ to a tensor that encodes high-order geometrical relations. However, a small increment in the order of relations leads to a combinatorial explosion of the amount of data needed to support the algorithm. Therefore, most of high-order GM methods can only work on very sparse graphs with no more than 3-order features. In addition, it is unclear on how to extend high-order methods to incorporate non-rigid deformations. Third, the performance of GM methods in real applications is often limited by the initial construction of graphs. To resolve this issue, Cho and Lee [42] proposed a progressive framework which combines probabilistic progression of graphs with matching of graphs. The algorithm re-estimates in a Bayesian manner the most plausible target graphs based on the current matching result, and guarantees to boost the matching objective at the subsequent graph matching.

## 2.3 Previous work on ICP

### 2.3.1 Definition of ICP

Given two sets of points, $\mathbf{P}_1 = [\mathbf{p}_1^1, \cdots, \mathbf{p}_{n_1}^1] \in \mathbb{R}^{d \times n_1}$ and $\mathbf{P}_2 = [\mathbf{p}_1^2, \cdots, \mathbf{p}_{n_2}^2] \in \mathbb{R}^{d \times n_2}$, iterative closest point (ICP) algorithms (*e.g.*, [21, 219]) aim to find the correspondence and the geometric transformation between points such that the sum of distances is minimized:

$$\min_{\mathbf{X} \in \Pi, \mathcal{T} \in \Psi} J_{icp}(\mathbf{X}, \mathcal{T}) = \sum_{i_1 i_2} x_{i_1 i_2} \|\mathbf{p}_{i_1}^1 - \tau(\mathbf{p}_{i_2}^2)\|_2^2 + \psi(\mathcal{T}), \tag{2.5}$$

where $\mathbf{X} \in \{0,1\}^{n_1 \times n_2}$ denotes the correspondence between points. Depending on the problem, $\mathbf{X}$ denotes either a one-to-one or many-to-one matching. In this chapter, we consider a one-to-one matching between points and $\mathbf{X}$ is thus constrained to be a permutation matrix *i.e.*, $\mathbf{X} \in \Pi$, where $\Pi$ is defined as Eq. 2.2. $\tau(\cdot) : \mathbb{R}^d \to \mathbb{R}^d$ denotes a geometric transformation parameterized by $\mathcal{T}$. The transformation is softly penalized by the function $\psi(\cdot)$ and constrained within the set $\Psi$. The parameterization of $\tau(\cdot)$, $\mathcal{T}$, $\psi(\cdot)$ and $\Psi$ can vary according to the requirement of the problem. Fig. 2.2 lists the parameterization of three common transformations in 2-D space: similarity, affine and RBF non-rigid transformation.

**Similarity transformation:** Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2 \times n}$, its similarity transformation is computed as $\tau(\mathbf{p}) = s\mathbf{R}\mathbf{p} + \mathbf{t}$ or $\tau(\mathbf{P}) = s\mathbf{R}\mathbf{P} + \mathbf{t}\mathbf{1}_n^T$, where $s \in \mathbb{R}$, $\mathbf{R} \in \mathbb{R}^{2 \times 2}$ and $\mathbf{t} \in \mathbb{R}^2$ denote the scaling factor, rotation matrix and translation vector respectively. In addition, the rotation matrix $\mathbf{R} \in \Psi$ has to satisfy the constraints $\Psi = \{\mathbf{R} | \mathbf{R}^T\mathbf{R} = \mathbf{I}_2, |\mathbf{R}| = 1\}$.

**Affine transformation:** Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2 \times n}$, its affine transformation is computed as $\tau(\mathbf{p}) = \mathbf{V}\mathbf{p} + \mathbf{t}$ or $\tau(\mathbf{P}) = \mathbf{V}\mathbf{P} + \mathbf{t}\mathbf{1}_n^T$, where $\mathbf{V} \in \mathbb{R}^{2 \times 2}$ and $\mathbf{t} \in \mathbb{R}^2$ denote the affine matrix and translation vector respectively.

**RBF non-rigid transformation:** The RBF non-rigid transformation is one of the most popular non-rigid transformation widely used in image registration [47, 135]. The parameterization of the transformation depends on the choice of the basis points. In our experiments, the basis points were chosen as the nodes of the second graph, *i.e.*, $\mathbf{P}_2 = [\mathbf{p}_1^2, \cdots, \mathbf{p}_{n_2}^2] \in \mathbb{R}^{2 \times n_2}$. Given a point $\mathbf{p} \in \mathbb{R}^2$ or a point set $\mathbf{P} \in \mathbb{R}^{2 \times n}$, the transformation is computed as a displacement shifted from its initial position, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p} + \mathbf{W}\phi(\mathbf{p})$ or $\tau(\mathbf{P}) = \mathbf{P} + \mathbf{W}\mathbf{L}_p$, where $\mathbf{W} \in \mathbb{R}^{2 \times n_2}$ is the weight matrix and $\phi(\mathbf{p}) = [\phi_1(\mathbf{p}), \cdots, \phi_{n_2}(\mathbf{p})]^T \in \mathbb{R}^{n_2}$ denotes

| | Similarity | Affine | RBF Non-rigid |
|---|---|---|---|
| $\mathcal{T}$ | $s \in \mathbb{R}$<br>$\mathbf{R} \in \mathbb{R}^{2\times 2}$<br>$\mathbf{t} \in \mathbb{R}^2$ | $\mathbf{V} \in \mathbb{R}^{2\times 2}$<br>$\mathbf{t} \in \mathbb{R}^d$ | $\mathbf{W} \in \mathbb{R}^{2\times n}$ |
| $\tau(\mathbf{p})$ | $s\mathbf{R}\mathbf{p} + \mathbf{t}$ | $\mathbf{V}\mathbf{p} + \mathbf{t}$ | $\mathbf{p} + \mathbf{W}\phi(\mathbf{p})$ |
| $\tau(\mathbf{P})$ | $s\mathbf{R}\mathbf{P} + \mathbf{t}\mathbf{1}_n^T$ | $\mathbf{V}\mathbf{P} + \mathbf{t}\mathbf{1}_n^T$ | $\mathbf{P} + \mathbf{W}\mathbf{L}_p$ |
| $\psi(\mathcal{T})$ | 0 | 0 | $\lambda_w \operatorname{tr}(\mathbf{W}\mathbf{L}_p\mathbf{W}^T)$ |
| $\Psi$ | $\mathbf{R}^T\mathbf{R} = \mathbf{I}_2$<br>$\lvert\mathbf{R}\rvert = 1$ | $\emptyset$ | $\emptyset$ |

Table 2.2: Parameterization of three geometrical transformations in 2-D space, where $\mathcal{T}$, $\tau(\cdot)$, $\Psi$ and $\psi(\cdot)$ denote the parameter set, transformation function, constraint set and penalization function respectively.

the $n_2$ displacement functions corresponding to the basis points. Each displacement function, $\phi_i(\mathbf{p}) = \exp(-\frac{\|\mathbf{p}-\mathbf{p}_i^2\|_2^2}{\sigma_w^2})$, is computed as a RBF between $\mathbf{p}$ and the basis $\mathbf{p}_i^2$ with the bandwidth $\sigma_w$. $\mathbf{L}_p = [\phi(\mathbf{p}_1^2), \cdots, \phi(\mathbf{p}_{n_2}^2)] \in \mathbb{R}^{n_2\times n_2}$ is the RBF kernel defined on all the pairs of basis points. Following [135], we regularize the transformation by penalizing the non-smoothness of the displacement field defined by $\mathbf{W}$. More specifically, the regularization term is computed as, $\psi(\mathcal{T}) = \lambda_w \operatorname{tr}(\mathbf{W}\mathbf{L}_p\mathbf{W}^T)$, where $\lambda_w \geq 0$ controls the trade-off between the objective and the regularization term.

### 2.3.2 Advances on ICP

In the past decade, ICP has been widely used to solve correspondence problems in image, shape, and surface registration [190, 229] due to its simplicity and low computational complexity. ICP methods can be broadly classified by the type of deformation they recover as rigid [21, 219] or non-rigid [47, 135].

A major limitation of ICP methods is that they are highly sensitive to the initialization. There have been various strategies proposed to address this issue [160]. One common choice is to in-

troduce structural information into the registration schemes [30, 64, 129]. For instance, Belongie *et al.*. [17] proposed shape context, a robust shape representation for finding correspondence between deformed shapes. Advanced optimization scheme can also be employed to improve the performance of ICP. For instance, Chui and Rangarajan [47] proposed to combine soft-assignment [152] with deterministic annealing for non-rigid point registration. This work has been recently extended in [135] for both rigid and non-rigid registration. The most closely related work to our method is the statistical approach [52, 127, 220], where a binary neighborhood graph is used for guiding the ICP minimization for finding better correspondence. Unlike existing ICP algorithms, the proposed FGM introduces pairwise constraints that makes the matching problem less prone to local minima.

Alternatively, RANSAC [71] type of algorithms have been widely used to find reliable correspondences between two or more images in the presence of outliers. RANSAC estimates a global relation that fits the data, while simultaneously classifying the data into inliers and outliers. Unlike RANSAC-type of algorithms, FGM is able to efficiently model non-rigid transformations and large rigid transformations.

## 2.4 Factorized graph matching (FGM)

This section proposes a novel factorization of the pairwise affinity matrix $\mathbf{K}$. As we will see in the following sections, this factorization provides a light-weight representation for GM problems, allows a unification of GM methods, elaborates a better optimization strategy and makes it easy to add geometric constraints to GM. This factorization is the main contribution of the chapter.

The pairwise affinity matrix $\mathbf{K}$ plays a central role in GM because it encodes all the first-order and second-order relations between graphs. Two properties of $\mathbf{K}$, full-rankness and indefiniteness, pose key challenges to conventional GM methods such as spectral methods [51, 112] and gradient-based ones [43, 115, 191, 216]. Fig. 2.3 illustrates an empirical study on a thousand of $\mathbf{K}$s computed from three realistic benchmark datasets in Section 2.8. The first row of Fig. 2.3 shows that the ratio between the rank and the dimension of $\mathbf{K}$ equaled to one, *i.e.*, $\mathbf{K}$ was a full-rank matrix in all the experiments. This fact explains why the spectral methods [51, 112]

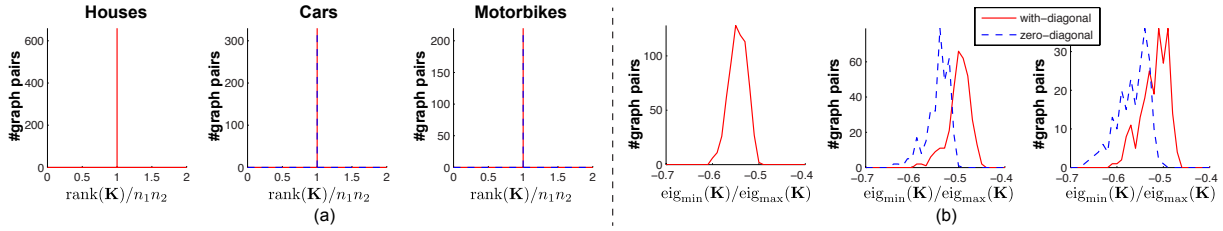


Figure 2.3: Statistics of $\mathbf{K}$ computed for the graphs extracted from three real image datasets used in the experiments. The histograms in the top row illustrate the ratio between the rank and the dimension of $\mathbf{K}$. The histograms in the bottom row show the ratio between the minimum and the maximum eigenvalues of the $\mathbf{K}$. For the last two datasets (columns), we test $\mathbf{K}$ with zero values on the diagonal as well. The top and bottom rows indicate that all the computed $\mathbf{K}$ are full-rank and indefinite respectively.

usually perform worse than others. This is because the spectral methods employ the rank-one approximation of $\mathbf{K}$ which inevitably leads to loss in accuracy. The second row of Fig. 2.3 shows that $\mathbf{K}$ was an indefinite matrix because the ratio between its maximum and minimum eigenvalues was smaller than $-0.4$ in the experiments. The indefiniteness of $\mathbf{K}$ indicates that the maximization of $\mathrm{vec}(\mathbf{X})^T\mathbf{K}\,\mathrm{vec}(\mathbf{X})$ is a non-convex problem and thus the gradient-based methods [43, 115, 191, 216] is prone to local optima. Instead of treating $\mathbf{K}$ as a black box, we analyze and propose a factorization that decouples the structure of $\mathbf{K}$. To the best of our knowledge, this work is the first to propose a closed-form factorization for $\mathbf{K}$ and its applications to GM problems.

To illustrate the intuition behind the factorization, let us revisit the synthetic problem shown in Fig. 2.2. Notice that $\mathbf{K} \in \mathbb{R}^{n_1n_2\times n_1n_2}$ (Fig. 2.2f) is composed by two types of affinities: the node affinity ($\mathbf{K}_p$) on its diagonal and the pairwise edge affinity ($\mathbf{K}_q$) on its off-diagonals. Without the diagonal, $\mathbf{K}$ is a sparse block matrix with three unique structures: (1) $\mathbf{K}$ is composed by $n_2$-by-$n_2$ smaller blocks $\mathbf{K}_{ij} \in \mathbb{R}^{n_1\times n_1}$. (2) Some of the $\mathbf{K}_{ij}$s are empty if there is no edge connecting the $i^{th}$ and $j^{th}$ nodes of $\mathcal{G}_2$. In other words, these empty blocks can be indexed by $\mathbf{G}_2\mathbf{H}_2^T$, *i.e.*, $\mathbf{K}_{ij} = \mathbf{0}$ if $[\mathbf{G}_2\mathbf{H}_2^T]_{ij} = 0$. (3) For the non-empty blocks, $\mathbf{K}_{ij}$ can be computed in a closed form as $\mathbf{G}_1\,\mathrm{diag}(\mathbf{k}_c^q)\mathbf{H}_1^T$, where $c$ is the index of the edge connecting the $i^{th}$ and $j^{th}$ nodes of $\mathcal{G}_2$, *i.e.*, $g_{ic}^2 = h_{jc}^2 = 1$. Based on these three observations, and after some linear algebra, it can

be shown that $\mathbf{K}$ can be factorized exactly as:

$$\mathbf{K} = \mathrm{diag}(\mathrm{vec}(\mathbf{K}_p)) + (\mathbf{G}_2 \otimes \mathbf{G}_1)\,\mathrm{diag}(\mathrm{vec}(\mathbf{K}_q))(\mathbf{H}_2 \otimes \mathbf{H}_1)^T. \tag{2.6}$$

This factorization decouples the graph structure ($\mathbf{G}_1$, $\mathbf{H}_1$, $\mathbf{G}_2$ and $\mathbf{H}_2$) from the similarity ($\mathbf{K}_p$ and $\mathbf{K}_q$). It is important to notice that our factorization significantly differs from the one proposed in our previous work [222] in two aspects: (1) Eq. 2.6 is proposed for more general graphs composed by directed edges while [222] can be only applied for simpler graphs with undirected edges; (2) Unlike [222], Eq. 2.6 separates $\mathbf{K}_p$ and $\mathbf{K}_q$ in two independent terms. This separation enables us to introduce geometric transformations on $\mathbf{K}_p$ and $\mathbf{K}_q$ for GM problems.

Eq. 2.6 is the key contribution of this work. Previous work on GM explicitly computed the computationally expensive (in space and time) $\mathbf{K}$, which is of size $O(n_1^2 n_2^2)$. On the contrary, Eq. 2.6 offers an alternative framework by replacing $\mathbf{K}$ with six smaller matrices, which are of size $O(n_1 m_1 + n_2 m_2 + n_1 n_2 + m_1 m_2)$. For instance, plugging Eq. 2.6 into Eq. 2.3 leads to an equivalent objective function:

$$J_{gm}(\mathbf{X}) = \mathrm{tr}\left(\mathbf{K}_p^T \mathbf{X}\right) + \mathrm{tr}\left(\mathbf{K}_q^T (\underbrace{\mathbf{G}_1^T \mathbf{X} \mathbf{G}_2 \circ \mathbf{H}_1^T \mathbf{X} \mathbf{H}_2}_{\mathbf{Y}})\right), \tag{2.7}$$

where $\mathbf{Y} \in \{0,1\}^{m_1 \times m_2}$ can be interpreted as a correspondence matrix for edges, *i.e.*, $y_{c_1 c_2} = 1$ if both nodes of $c_1^{th}$ edge in $\mathcal{G}_1$ are matched to the nodes of $c_2^{th}$ edge in $\mathcal{G}_2$. For instance, Fig. 2.2d illustrates the node and edge correspondence matrices for the matching defined in Fig. 2.2a.

## 2.5 A unified framework for graph matching

In past decades, a myriad of GM methods have been proposed for solving a variety of applications. Previous GM methods varied in their objective formulations and optimization strategies. Although much effort [32, 36, 48, 166] has been made on comparing GM methods, a mathematical framework to connect the various GM methods in a coherent manner is lacking. This section derives a unified framework to cast many GM methods using our proposed factorization.

Beyond the unification of GM methods, we show that a close relation exists between GM and ICP objectives. This insight allows us to augment GM problems with additional geometrical constraints that are necessary in many computer vision applications. Moreover, we extend ICP

to incorporate pair-wise constraints. Finally, we show (using the factorization) the connection between GM and Markov random field (MRF) methods.

### 2.5.1 Unification of GM methods

Eq. 2.3 summarizes one of the most important GM problems studied in recent research on computer vision [43, 51, 62, 77, 112, 115, 165, 180, 183, 184, 191, 216]. However, the way of formulating GM is not unique. In other applications [9, 35, 49, 144, 188, 214], the goal of GM was alternatively formulated as:

$$\max_{\mathbf{X}\in\Pi} \quad \mathrm{tr}(\mathbf{K}_p^T\mathbf{X}) + \mathrm{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T), \tag{2.8}$$

where the main difference from Eq. 2.3 is in the second term, which measures the edge compatibility as the linear similarity between two weighted adjacency matrices $\mathbf{A}_1 \in [0,1]^{n_1\times n_1}$ and $\mathbf{A}_2 \in [0,1]^{n_2\times n_2}$, given the permutation $\mathbf{X}$. In this case, the topology of a graph is defined by a weighted adjacency matrix, $\mathbf{A} \in [0,1]^{n\times n}$, where each element, $a_{ij}$, indicates the soft connectivity between the $i^{th}$ and the $j^{th}$ nodes.

GM can be formulated as the classical QAP, that has been well studied in the literature of operation research since the $40's$. In operation research literature [124], Eq. 2.3 and Eq. 2.8 are known as Lawler's QAP [110] and Koopmans-Beckmann's QAP [105] respectively. Please refer to Fig. 2.1 for a taxonomy of previous GM works in computer vision from a QAP perspective. Roughly speaking, Lawler's QAP is more general than Koopmans-Beckmann's QAP because it has $\frac{1}{2}n_1^2n_2^2$ free variables in $\mathbf{K}$ compared to $\frac{1}{2}n_1^2 + \frac{1}{2}n_2^2$ free variables in $\mathbf{A}_1$ and $\mathbf{A}_2$. In fact, Koopmans-Beckmann's QAP can always be represented as a special case of Lawler's QAP if we assume $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$. In this particular Lawler's QAP, the edge feature has to be a 1-D scalar (*i.e.*, $\mathbf{q}_c = a_{ij}$) and the edge similarity has to be linear (*i.e.*, $\kappa^q_{c_1c_2} = \langle \mathbf{q}^1_{c_1}, \mathbf{q}^2_{c_2}\rangle$). This special QAP can be limited when representing the challenging matching problem encountered in real cases. However, Lawler's QAP considers more general similarity measures including non-linear Gaussian kernels, which take the linear similarity as an instance.

In general, it is unclear on how to understand the commonalities and differences between these two types of GMs. In the following, we will propose a simple and clean connection that exists using the proposed factorization. Observe that $\mathbf{K}_q$ can always be factorized (*e.g.*, SVD) as

$\mathbf{K}_q = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{m_1 \times c}$ and $\mathbf{V} \in \mathbb{R}^{m_2 \times c}$. Taking advantage of the low-rank structure of $\mathbf{K}_q$, Eq. 2.7 can be re-formulated[2] as follows:

$$\begin{aligned}
& J_{gm}(\mathbf{X}) \\
&= \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \mathrm{tr}\left((\mathbf{U}\mathbf{V}^T)^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\right) \\
&= \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \mathrm{tr}\left((\sum_{i=1}^{c}\mathbf{u}_i\mathbf{v}_i^T)^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\right) \\
&= \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \sum_{i=1}^{c}\mathrm{tr}\left(\mathrm{diag}(\mathbf{u}_i)\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2\,\mathrm{diag}(\mathbf{v}_i)\mathbf{H}_2^T\mathbf{X}^T\mathbf{H}_1\right) \\
&= \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \sum_{i=1}^{c}\mathrm{tr}\left(\mathbf{A}_i^1\mathbf{X}\mathbf{A}_i^2\mathbf{X}^T\right), \qquad (2.9)
\end{aligned}$$

where $\mathbf{A}_i^1 = \mathbf{G}_1\,\mathrm{diag}(\mathbf{u}_i)\mathbf{H}_1^T \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{A}_i^2 = \mathbf{G}_2\,\mathrm{diag}(\mathbf{v}_i)\mathbf{H}_2^T \in \mathbb{R}^{n_2 \times n_2}$ can be interpreted as adjacency matrices for the two graphs. Eq. 2.9 reveals that $c$, the rank of the edge affinity matrix $\mathbf{K}_q$, is a key characteristic of GM methods. If $c = 1$, Lawler's QAP downgrades to the Koopmans-Beckmann's QAP. In other cases when $c > 1$, solving Lawler's QAP can be cast as a joint optimization of $c$ Koopmans-Beckmann's QAPs.

Despite its importance, the relation between Eq. 2.3 and Eq. 2.8 has been rarely explored in the literature of GM. In fact, many GM methods can benefit from this connection. Consider the special case when the node affinity matrix $\mathbf{K}_p$ is empty and the edge affinity matrix $\mathbf{K}_q = \mathbf{u}\mathbf{v}^T$ has a rank-1 structure. According to the factorization, we can conclude $\mathbf{K} = \mathbf{A}_2 \otimes \mathbf{A}_1$, where $\mathbf{A}_1 = \mathbf{G}_1\,\mathrm{diag}(\mathbf{u})\mathbf{H}_1^T$ and $\mathbf{A}_2 = \mathbf{G}_2\,\mathrm{diag}(\mathbf{v})\mathbf{H}_2^T$. In this case, the spectral matching algorithm [112] can be more efficiently computed as $\mathrm{eig}(\mathbf{K}) = \mathrm{eig}(\mathbf{A}_2) \otimes \mathrm{eig}(\mathbf{A}_1)$, where $\mathrm{eig}(\mathbf{K})$ denotes the leading eigen-vector of $\mathbf{K}$. In addition, we can use Umeyama's spectral algorithm [188] to find the approximate solution by maximizing $\mathrm{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$ subject to $\mathbf{X}\mathbf{X}^T = \mathbf{I}$.

### 2.5.2 Connections between GM and ICP

To connect ICP with GM methods, we denote the node affinity matrix as, $\mathbf{K}_p(\mathcal{T}) \in \mathbb{R}^{n_1 \times n_2}$, where each element, $\kappa_{i_1 i_2}^p(\mathcal{T}) = -\|\mathbf{p}_{i_1}^1 - \tau(\mathbf{p}_{i_2}^2)\|_2^2$, encodes the negative Euclidean distance

[2]The equation, $\mathrm{tr}((\mathbf{u}\mathbf{v}^T)^T(\mathbf{A} \circ \mathbf{B})) = \mathrm{tr}(\mathrm{diag}(\mathbf{u})\mathbf{A}\,\mathrm{diag}(\mathbf{v})\mathbf{B}^T)$, always holds for arbitrary $\mathbf{u} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^n$ and $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$.

between nodes. Using this short notation, the original ICP objective (Eq. 2.5) can be concisely reformulated as:

$$J_{icp}(\mathbf{X}, \mathcal{T}) = -\operatorname{tr}\left(\mathbf{K}_p(\mathcal{T})^T\mathbf{X}\right) + \psi(\mathcal{T}). \tag{2.10}$$

Eq. 2.10 reveals a clean connection between the ICP and GM objective (Eq. 2.7). Given the transformation ($\mathcal{T}$), minimizing $J_{icp}$ over $\mathbf{X}$ is equivalent to maximize the node compatibility in Eq. 2.7. This optimization can be cast as a linear matching problem, which can be efficiently optimized by the Hungarian algorithm if $\mathbf{X}$ is a one-to-one mapping or the winner-take-all manner if $\mathbf{X}$ is a many-to-one mapping. In general, however, the joint optimization over $\mathbf{X}$ and $\mathcal{T}$ is non-convex, and no-closed form solution is known. Typically, some sort of alternated minimization (*e.g.*, EM, coordinate-descent) is needed to find a local optima.

### 2.5.3 Connections between GM and MRF

Beyond GM problems, pairwise constraints have been popular in formulating other types of matching problems. Among them, Markov random field (MRF) is perhaps the most well-studied one and it plays a key role in obtaining binary solutions to many applications [179] such as stereo, image stitching and segmentation.

Given $n_1$ nodes and $n_2$ labels, MRF problems consist in finding the optimal labeling matrix $\mathbf{X} \in \Phi$ that defines a many-to-one mapping from nodes to labels, *i.e.*:

$$\Phi = \{\mathbf{X} | \mathbf{X} \in \{0,1\}^{n_1 \times n_2}, \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}\}. \tag{2.11}$$

For instance, Fig. 2.4 illustrates a simple MRF problem, where four nodes need to be assigned to three labels.

The goal of MRF problems often reduces to optimizing an energy function that depends on how well the labels agree with the nodes (first-order potentials) as well as on how well pairs of labels at connected nodes agree with each other (second-order potentials). Similar to GM, all the relations between nodes and labels can be summarized in a pairwise affinity matrix, $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$, whose diagonal and off-diagonal elements encode the first-order and second-order potentials respectively. Following the previous work [50, 113, 155], the general MRF can be

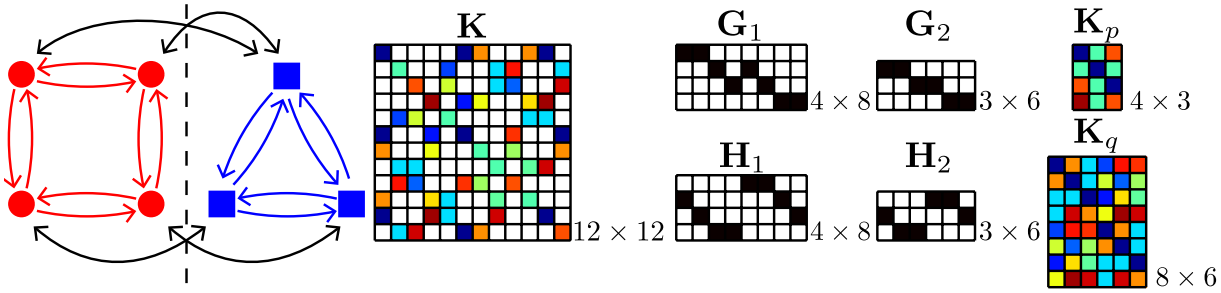


Figure 2.4: An MRF example. Labeling four nodes (red circles) with three labels (blue boxes). Similar to GM, the pairwise affinity matrix $\mathbf{K}$ can be factorized into six smaller matrices.

formalized as a similar QAP as follows:

$$\max_{\mathbf{X}\in\Phi} \quad J_{mrf}(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^T \mathbf{K}\, \mathrm{vec}(\mathbf{X}), \tag{2.12}$$

Two general differences exist between GM and MRF. First, GM finds a one-to-one mapping (Eq. 2.2) between the nodes of two graphs. In GM, the order of the two graphs is usually not important because the permutation constraint (Eq. 2.2) is symmetric. However, MRF assigns labels to nodes and it optimizes an asymmetrical many-to-one mapping (Eq. 2.11) between nodes and labels. Each node has to be associated with only one label, while each label is allowed to accept multiple nodes. Second, the two graphs to be matched in GM are used to describe two similar instances. In practice, the nodes in both graphs are sparsely connected. However, MRF matches between two types of graphs, the graph of nodes and the graph of labels. Consider the task of labeling four image pixels in Fig. 2.4 as an example. On one hand, the node graph is a sparse planar grid of image pixels, where eight edges connect neighbor pixels. On the other hand, the label graph is fully connected by six edges. This is because the labels are three independent states. Any combination of two labels is possible to label a pair of pixels.

Although it is originally designed for GM problems, our factorization (Eq. 2.6) can be applied to model and understand MRF problems as well. For instance, the 12-by-12 $\mathbf{K}$ modeling the problem shown in Fig. 2.4 can be decomposed into six components. In particular, $\mathbf{G}_1$ and $\mathbf{H}_1$ are two 4-by-8 binary matrices, encoding the node-edge incidence of the node graph. $\mathbf{G}_2$ and $\mathbf{H}_2$ are two 3-by-6 binary ones, describing the fully connected label graph. $\mathbf{K}_p$ and $\mathbf{K}_q$ are used to encode the first-order and second-order potentials respectively.

Despite the similarity between MRFs and GMs, some of the methods originally designed for MRFs (*e.g.*, graph cuts, belief propagation, linear programming and dual decomposition) cannot be directly extended for solving GMs:

**Graph cuts and belief propagation.** Graph cuts [25, 103] and belief propagation [203] are among the most popular tools for approximating MRFs. Due to their quadratic complexity in the number of discrete states, however, neither graph cuts nor belief propagation is suitable for solving practical GMs problems, where the state number of each node (*i.e.*, number of nodes in the other graph) can be much larger.

**Linear programming.** Linear programming (LP) methods [204, 210] approximate MRFs by introducing a new variable $\mathbf{Y} = \mathrm{vec}(\mathbf{X})^T \mathrm{vec}(\mathbf{X})$ to rewrite MRF's objective (Eq. 2.12) in a linear form. As shown in Fig. 2.5, however, such reformulation often leads to a quadratic explosion of the variable and constraints. And even the most advanced LP solver (*e.g.*, CPLEX [145]) can only handle very small graphs.

**Dual decomposition.** Instead of optimizing the primal objective of MRFs, some methods [102, 195] turns to solve the dual because of the theoretical guarantee in convergence. In particular, Komodakis *et al.* [104] introduced a dual-decomposition (DD) [20] framework that yields optimal solution in many practical computer vision problems [98]. More recently, Torresani *et al.* [185] applied the similar DD idea to GMs by decomposing original problem into several easier sub-problems. Despite its accuracy in approximation, this method still suffers from some practical issues when dealing with GMs. For instance, the graph used in [185] is very sparse because the complexity of solving each sub-problems is $O(nm^n)$ where $n$ and $m$ are the numbers of nodes and edges of decomposition respectively. In addition, the performance of DD is largely affected by the quality of the decomposition of the original problem.

## 2.6 A path-following algorithm

This section presents a path-following algorithm for approximating the GM problem. Traditionally, Eq. 2.3 is approached by a two-step scheme: (1) solving a continuously relaxed problem and (2) rounding the approximate solution to a binary one. This procedure has at least two lim-

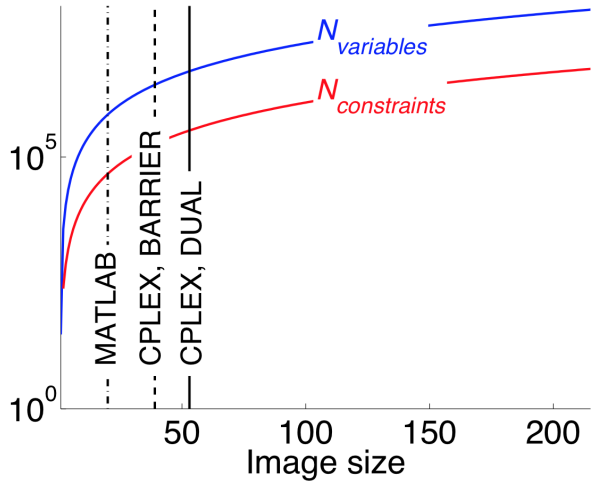


Figure 2.5: The number of variables and constraints in the LP relaxation of the stereo disparity problem as a function of the size of the image. Image was taken from [210].

itations. First, the continuous relaxation is non-convex and prone to local optima. Second, the rounding step will inevitably cause accuracy loss because it is independent of the cost function. Inspired by [128, 137, 214], we address these two issues using a path-following algorithm by iteratively optimizing an interpolation of two relaxations. This new scheme has three theoretical advantages: (1) The optimization performance is initialization-free; (2) The final solution is guaranteed to converge to an integer one and therefore no rounding step is needed; (3) The iteratively updating procedure resembles the idea of numerical continuation methods [8], which have been successfully used for solving nonlinear systems of equations for decades.

### 2.6.1 Convex and concave relaxation

To employ the path-following algorithm, we need to find a convex and concave relaxation of $J_{gm}(\mathbf{X})$. Fortunately, the factorization (Eq. 2.6) offers a simple and principal way to derive these approximations. To do that, let's first introduce an auxiliary function:

$$J_{con}(\mathbf{X}) = \sum_{i=1}^{c} \mathrm{tr}\left(\mathbf{A}_i^{1^T}\mathbf{X}\mathbf{X}^T\mathbf{A}_i^1\right) + \mathrm{tr}\left(\mathbf{A}_i^2\mathbf{X}^T\mathbf{X}\mathbf{A}_i^{2^T}\right).$$

As we know, a permutation matrix[3] $\mathbf{X} \in \Pi$ is also an orthogonal one, satisfying $\mathbf{X}\mathbf{X}^T = \mathbf{X}^T\mathbf{X} = \mathbf{I}$. Therefore, $J_{con}(\mathbf{X}) = \gamma$ is always a constant,

$$\gamma = \sum_{i=1}^{c} \mathrm{tr}\left(\mathbf{A}_i^{1^T}\mathbf{A}_i^1\right) + \mathrm{tr}\left(\mathbf{A}_i^2\mathbf{A}_i^{2^T}\right),$$

if $\mathbf{X}$ is a permutation matrix.

Introducing into $J_{gm}(\mathbf{X})$ the constant term, $\frac{1}{2}J_{con}(\mathbf{X})$, yields two equivalent objectives:

$$J_{vex}(\mathbf{X}) = J_{gm}(\mathbf{X}) - \frac{1}{2}J_{con}(\mathbf{X}) = \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) - \frac{1}{2}\sum_{i=1}^{c}\|\mathbf{X}^T\mathbf{A}_i^1 - \mathbf{A}_i^2\mathbf{X}^T\|_F^2, \tag{2.13}$$

$$J_{cav}(\mathbf{X}) = J_{gm}(\mathbf{X}) + \frac{1}{2}J_{con}(\mathbf{X}) = \mathrm{tr}\left(\mathbf{K}_p^T\mathbf{X}\right) + \frac{1}{2}\sum_{i=1}^{c}\|\mathbf{X}^T\mathbf{A}_i^1 + \mathbf{A}_i^2\mathbf{X}^T\|_F^2. \tag{2.14}$$

The above two functions achieve the same value up to a constant difference $\frac{\gamma}{2}$ as $J_{gm}(\mathbf{X})$ with respect to any permutation and orthogonal matrix. However, the orthogonal constraint $\mathbf{X}^T\mathbf{X} = \mathbf{I}$ on $\mathbf{X}$ is not a convex constraint. Therefore, we further relax $\mathbf{X}$ to be a doubly-stochastic matrix, $\mathbf{X}\mathbf{1} \leq \mathbf{1}, \mathbf{X}^T\mathbf{1} = \mathbf{1}$. Under this constraint, the problems of maximizing $J_{vex}(\mathbf{X})$ and $J_{cav}(\mathbf{X})$ are convex and concave respectively. This is because their Hessians,

$$\nabla^2_{\mathbf{X}} J_{vex}(\mathbf{X}) = -\sum_{i=1}^{c}(\mathbf{I}\otimes\mathbf{A}_i^1 - \mathbf{A}_i^2\otimes\mathbf{I})^T(\mathbf{I}\otimes\mathbf{A}_i^1 - \mathbf{A}_i^2\otimes\mathbf{I}),$$

$$\nabla^2_{\mathbf{X}} J_{cav}(\mathbf{X}) = \sum_{i=1}^{c}(\mathbf{I}\otimes\mathbf{A}_i^1 + \mathbf{A}_i^2\otimes\mathbf{I})^T(\mathbf{I}\otimes\mathbf{A}_i^1 + \mathbf{A}_i^2\otimes\mathbf{I}),$$

are always negative and positive semi-definite, respectively.

### 2.6.2 A path-following strategy

The main challenge of approximating GM comes from its non-convex objective and the combinatorial constraints. Inspired by [128, 137, 214], we optimize Eq. 2.3 by iteratively optimizing a series of the following convex-concave problems [213]:

$$\max_{\mathbf{X}\in\mathcal{D}} \quad J_\alpha(\mathbf{X}) = (1-\alpha)J_{vex}(\mathbf{X}) + \alpha J_{cav}(\mathbf{X}), \tag{2.15}$$

where $\alpha \in [0, 1]$ is a tradeoff between the convex relaxation and the concave one.

The advantages of the path-following algorithm over conventional GM algorithms are threefold: (1) The algorithm starts with a convex problem when $\alpha = 0$. Because the problem is

[3]We need to introduce dummy selection variables if the two graphs are of different sizes.

convex, a numerical optimizer can find the global optimal solution independently of the initialization. (2) When $\alpha = 1$ (at the end of the iterations) the problem is concave. It has been well-studied [87, 157] that any local optima of a concave optimization problem must lie at the extreme point of the constraint set. According to the Birkhoff-von Neumann theorem [193], all the doubly-stochastic matrices ($\mathbf{X} \in \mathcal{D}$) form the Birkhoff polytope, whose vertices are precisely the permutation matrices ($\mathbf{X} \in \Pi$). Therefore, the converged solution is guaranteed to be binary and no further discrete rounding step is needed. (3) By smoothly increasing $\alpha$ from $\alpha = 0$ to $\alpha = 1$, the path-following algorithm is more likely to find better local optima than gradient-based method.

To have a better understanding, we provide an example of optimizing GM using this strategy in Fig. 2.6. The graphs to be matched are extracted from one pair of car images used in the experiment (See Section 2.8.4). Fig. 2.6a plots the real GM objective ($J_{gm}$) and the objective ($J_\alpha$) to be optimized with respect to the change of $\alpha$. The convex ($J_{vex}$) and concave ($J_{cav}$) components are also plotted. Overall, four observations can be concluded looking at this figure. First, $J_\alpha$ equals to $J_{vex}$ and $J_{cav}$ when $\alpha = 0$ and $\alpha = 1$ respectively. Second, the curves of $J_{vex}$ and $J_{cav}$ are monotonically decreasing and increasing as $\alpha \to 1$. This is because as $\alpha$ increases, the concave part gets more control than the concvex part to direct the optimization of $J_\alpha$. Third, the curve of $J_\alpha$ intersects $J_{gm}$ at $\alpha = \frac{1}{2}$. This is true due to the fact that the following equation holds:

$$J_\alpha(\mathbf{X}) = J_{gm}(\mathbf{X}) + (\alpha - \frac{1}{2})J_{con}(\mathbf{X}), \tag{2.16}$$

by plugging Eq. 2.13 and Eq. 2.14 into Eq. 2.15. At last, when the algorithm converges at $\alpha = 1$, the gap between $J_{gm}$ and the two relaxations is a constant $\frac{\gamma}{2}$. This is because $\mathbf{X}$ turns to be a permutation matrix (Fig. 2.6c) and the equation, $J_{con}(\mathbf{X}) = \gamma$, always holds at $\alpha = 1$.

### 2.6.3 Sub-problem optimization

For a specific $\alpha$, we optimize $J_\alpha(\mathbf{X})$ using the Frank-Wolfe's algorithm (FW) [73, 92], a simple yet powerful method for constrained nonlinear programming. Unlike gradient-based methods that require a projection onto the constraint, FW algorithm efficiently updates the solution by automatically staying in the feasible set.

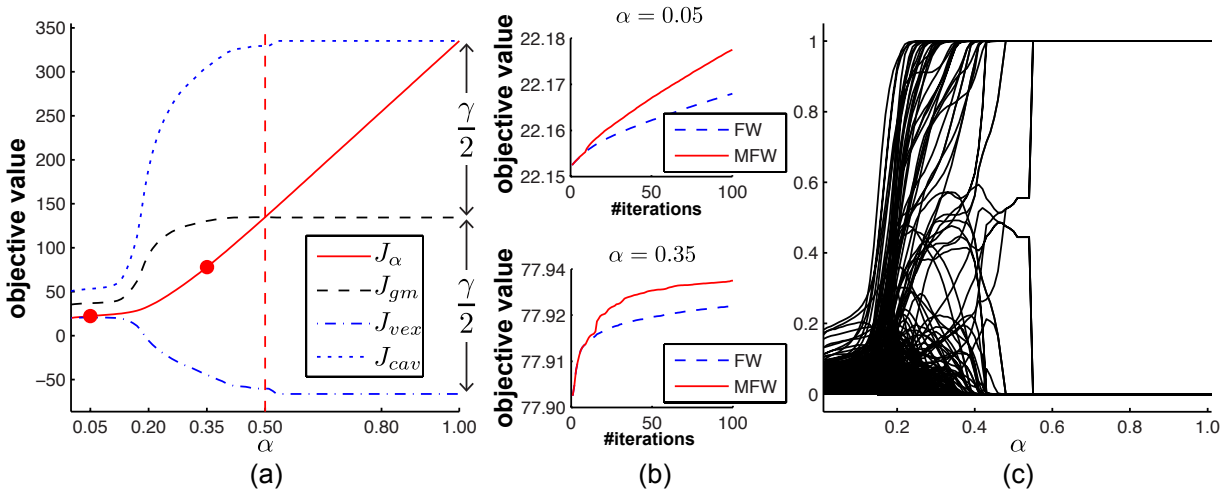


Figure 2.6: An example of using the path-following algorithm for optimizing the GM problem. (a) The comparison of the objectives during the optimization with respect to the change of $\alpha$. (b) The comparison of using FW and MFW for optimizing $J_\alpha(\mathbf{X})$ at two instances of $\alpha$. (c) The change of the elements of $\mathbf{X}$ as $\alpha \to 1$, where each curve corresponds a $x_{ij}$.

Given an initial $\mathbf{X}_0$, FW successively updates the solution as $\mathbf{X}^* = \mathbf{X}_0 + \eta\mathbf{Y}$ until converged. At each step, it needs to compute two components: the optimal direction $\mathbf{Y} \in \mathcal{D}$ and the optimal step size $\eta \in [0, 1]$.

To compute $\mathbf{Y}$, we solve the following linear programming problem:

$$\max_{\mathbf{Y}\in\mathcal{D}} \quad \mathrm{tr}\Big(\nabla J_\alpha(\mathbf{X}_0)^T(\mathbf{Y} - \mathbf{X}_0)\Big), \tag{2.17}$$

whose objective is the first-order approximation of $J_\alpha(\mathbf{X})$ at the point $\mathbf{X}_0$. The gradients $\nabla J_\alpha(\mathbf{X}_0)$ can be efficiently computed using matrix operation as follows:

$$\begin{aligned}
\nabla J_\alpha(\mathbf{X}) &= \nabla J_{gm}(\mathbf{X}) + (\alpha - \frac{1}{2})\nabla J_{con}(\mathbf{X}),\\
\nabla J_{gm}(\mathbf{X}) &= \mathbf{K}_p + \mathbf{H}_1(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{K}_q)\mathbf{H}_2^T + \mathbf{G}_1(\mathbf{H}_1^T\mathbf{X}\mathbf{H}_2 \circ \mathbf{K}_q)\mathbf{G}_2^T,\\
\nabla J_{con}(\mathbf{X}) &= 2\Big(\mathbf{G}_1(\mathbf{H}_1^T\mathbf{H}_1 \circ \mathbf{U}\mathbf{U}^T)\mathbf{G}_1^T + \mathbf{G}_2(\mathbf{H}_2^T\mathbf{H}_2 \circ \mathbf{V}\mathbf{V}^T)\mathbf{G}_2^T\Big)\mathbf{X}.
\end{aligned}$$

Similar to [115, 214], we adopt the Hungarian algorithm to efficiently solve this linear programming.

Once the gradient is computed, the line search for the optimal $\eta$ can be found in closed form.

More specifically, the optimal $\eta$ is the maximum point of the following parabola:

$$J_\alpha(\mathbf{X}_0 + \eta\mathbf{Y}) = \underbrace{a\eta^2 + b\eta}_{f(\eta)} + \text{const}, \tag{2.18}$$

where $a = a_{gm} + (\alpha - \frac{1}{2})a_{con}$ and $b = b_{gm} + (\alpha - \frac{1}{2})b_{con}$ are the $2^{nd}$ and $1^{st}$ order coefficients respectively. In addition, we can compute the components of $a_{gm}$, $b_{gm}$, $a_{con}$ and $b_{con}$ as follows:

$$\begin{aligned}
a_{gm} &= \operatorname{tr}\Big(\mathbf{K}_q^T(\mathbf{G}_1^T\mathbf{Y}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{Y}\mathbf{H}_2)\Big),\\
b_{gm} &= \operatorname{tr}\Big(\mathbf{K}_p^T\mathbf{Y}\Big) + \operatorname{tr}\Big(\mathbf{K}_q^T(\mathbf{G}_1^T\mathbf{X}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{Y}\mathbf{H}_2 + \mathbf{G}_1^T\mathbf{Y}\mathbf{G}_2 \circ \mathbf{H}_1^T\mathbf{X}\mathbf{H}_2)\Big),\\
a_{con} &= \operatorname{tr}\Big((\mathbf{U}\mathbf{U}^T)^T(\mathbf{G}_1^T\mathbf{Y}\mathbf{Y}^T\mathbf{G}_1 \circ \mathbf{H}_1^T\mathbf{H}_1)\Big) + \operatorname{tr}\Big((\mathbf{V}\mathbf{V}^T)^T(\mathbf{G}_2^T\mathbf{Y}\mathbf{Y}^T\mathbf{G}_2 \circ \mathbf{H}_2^T\mathbf{H}_2)\Big),\\
b_{con} &= 2\operatorname{tr}\Big((\mathbf{U}\mathbf{U}^T)^T(\mathbf{G}_1^T\mathbf{X}\mathbf{Y}^T\mathbf{G}_1 \circ \mathbf{H}_1^T\mathbf{H}_1)\Big) + 2\operatorname{tr}\Big((\mathbf{V}\mathbf{V}^T)^T(\mathbf{G}_2^T\mathbf{X}\mathbf{Y}^T\mathbf{G}_2 \circ \mathbf{H}_2^T\mathbf{H}_2)\Big).
\end{aligned}$$

It is well-known that the optimum point of the parabola must be one of the three points, $\eta^* \in \{0, 1, -\frac{b}{2a}\}$. A straightforward way to obtain the optimal $\eta^*$ is to compute $J_\alpha(\mathbf{X}_0 + \eta\mathbf{Y})$ at each point and pick the one yielding the largest value. In fact, it is more efficient to choose $\eta^*$ based on the geometry of the parabola. Fig. 2.7 illustrates the eight possible configurations of the parabola and the corresponding optimal step sizes.

### 2.6.4 Other implementation details

A similar path-following strategy was proposed in [214] and its performance over the state-of-the-art methods has been therein demonstrated for solving the less general GM problem (*i.e.*, $\operatorname{tr}(\mathbf{K}_p^T\mathbf{X}) + \operatorname{tr}(\mathbf{A}_1\mathbf{X}\mathbf{A}_2\mathbf{X}^T)$). After exhaustive testing this strategy for solving the most general GM problem (*i.e.*, $\operatorname{vec}(\mathbf{X})^T\mathbf{K}\operatorname{vec}(\mathbf{X})$), we empirically found that results can be improved using the following the following two steps:

**Convergence:** Although the FW algorithm is easy to implement, it converges sub-linearly. To get faster convergence speed while keeping its advantages in efficiency and low memory cost, we adopt a modified Frank-Wolfe (MFW) [74] to find a better searching direction $\mathbf{Y}$ by a convex combination of previously solutions. In the experiments, we used the directions computed in $10$ previous steps. Fig. 2.6b compares MFW with the original FW for two values of $\alpha$. As we can see, MFW converges much faster than FW.

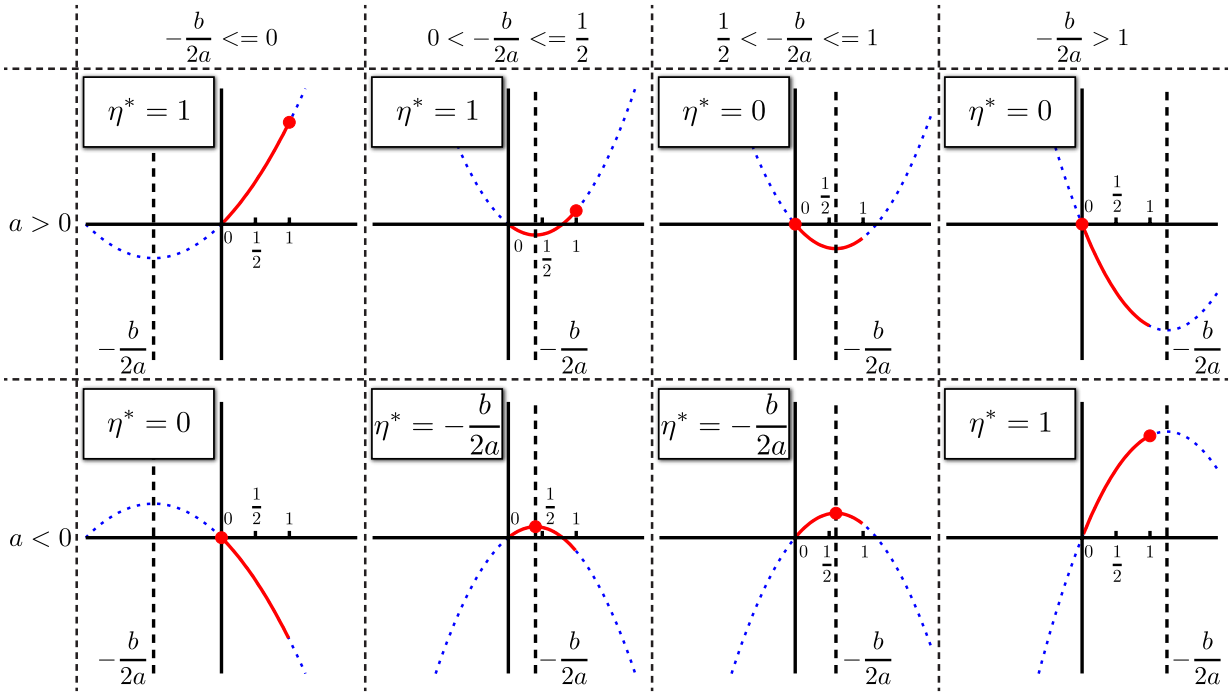


Figure 2.7: Eight possible shapes of the parabola $f(\eta) = a\eta^2 + b\eta$ and the corresponding optimal step sizes $\eta^* = \arg\max_{\eta \in [0,1]} f(\eta)$.

**Local vs global:** Although the path-following strategy returns an integer solution by smoothly tracking the local optima in a convex space, it does not guarantee to obtain the global optimal solution of the non-convex objective function. An important reason is that at each step, it locally optimizes over $J_\alpha(\mathbf{X})$ instead of the global one $J_{gm}(\mathbf{X})$. It is possible that $J_\alpha(\mathbf{X})$ increases while $J_{gm}(\mathbf{X})$ decreases. In order to escape from this phenomenon, we keep increasing the global score of $J_{gm}(\mathbf{X})$ during the optimization by discarding the bad temporary solution that worsens the score of $J_{gm}(\mathbf{X})$ and computing an alternative one by applying one step of FW for optimizing $J_{gm}(\mathbf{X})$. This refinement is analogous to the usage of FW in [115].

Algorithm 10 summarizes the work-flow of our algorithm. The initial $\mathbf{X}$ can be an arbitrary doubly stochastic matrix. The complexity of our algorithm can be roughly calculated as $O\big(T(\tau_{hun} + \tau_\nabla + \tau_\lambda) + \tau_{svd}\big)$, where $T$ is the number of iterations for the FW, and $\tau_{svd} = m_1 m_2^2$ is the cost of computing the SVD of $\mathbf{K}_q$. The Hungarian algorithm can be finished in $\tau_{hun} = \max(n_1^3, n_2^3)$. The gradient of $\nabla J_\alpha$ and the line search of $\lambda$ incur the same computational cost, $\tau_\nabla = \tau_\lambda = m_1 m_2$.

**Algorithm 1:** A path-following algorithm

**input** : $\mathbf{K}_p$, $\mathbf{K}_q$, $\mathbf{G}_1$, $\mathbf{H}_1$, $\mathbf{G}_2$, $\mathbf{H}_2$, $\delta$, $\mathbf{X}_0$

**output**: $\mathbf{X}$

*Initialize* $\mathbf{X}$ *to be a doubly stochastic matrix*;
*Factorize* $\mathbf{K}_q = \mathbf{U}\mathbf{V}^T$;
**for** $\alpha = 0 : \delta : 1$ **do** *Path-following*
    **if** $\alpha = 0.5$ *and* $J_{gm}(\mathbf{X}) < J_{gm}(\mathbf{X}_0)$ **then**
        *Update* $\mathbf{X} \leftarrow \mathbf{X}_0$;
    *Optimize Eq. 2.15 via MFW to obtain* $\mathbf{X}^*$;
    **if** $J_{gm}(\mathbf{X}^*) < J_{gm}(\mathbf{X})$ **then**
        *Optimize Eq. 2.3 via one step of FW to obtain* $\mathbf{X}^*$;
    *Update* $\mathbf{X} \leftarrow \mathbf{X}^*$;

## 2.7 Deformable graph matching (DGM)

GM has been widely used as a general tool in matching point sets with similar structures. In many computer vision applications, it is often required that the matching between two sets is constrained by a geometric transformation. It is unclear how to incorporate geometric constraints into GM methods. This section describes an extension of GM, called deformable graph matching (DGM), that incorporates rigid and non-rigid transformations into graph matching.

### 2.7.1 Objective function

To simplify the discussion and to be consistent with ICP, we compute the node feature of each graph $\mathcal{G} = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}, \mathbf{H}\}$ simply as the node coordinates, $\mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{d \times n}$. Similarly, the edge features $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_m] \in \mathbb{R}^{d \times m}$ are computed as the coordinate difference between the connected nodes, *i.e.*, $\mathbf{q}_c = \mathbf{p}_i - \mathbf{p}_j$, where $g_{ic} = h_{jc} = 1$. In this case, the edge feature can be conveniently computed in a matrix form as, $\mathbf{Q} = \mathbf{P}(\mathbf{G} - \mathbf{H})$.

Suppose that we are given two graphs, $\mathcal{G}_1 = \{\mathbf{P}_1, \mathbf{Q}_1, \mathbf{G}_1, \mathbf{H}_1\}$ and $\mathcal{G}_2 = \{\mathbf{P}_2, \mathbf{Q}_2, \mathbf{G}_2, \mathbf{H}_2\}$,

and a geometrical transformation defined on points by $\tau(\cdot)$. Similar to ICP, we compute the node affinity $\mathbf{K}_p(\mathcal{T}) \in \mathbb{R}^{n_1 \times n_2}$ and the edge affinity $\mathbf{K}_q(\mathcal{T}) \in \mathbb{R}^{m_1 \times m_2}$ as linear functions of the Euclidean distance[4], *i.e.*:

$$\begin{aligned} \kappa^p_{i_1 i_2}(\mathcal{T}) &= -\|\mathbf{p}^1_{i_1} - \tau(\mathbf{p}^2_{i_2})\|_2^2, \\ \kappa^q_{c_1 c_2}(\mathcal{T}) &= \beta - \| \underbrace{(\mathbf{p}^1_{i_1} - \mathbf{p}^1_{j_1})}_{\mathbf{q}^1_{c_1}} - \underbrace{(\tau(\mathbf{p}^2_{i_2}) - \tau(\mathbf{p}^2_{j_2}))}_{\tau(\mathbf{q}^2_{c_2})} \|_2^2, \end{aligned} \tag{2.19}$$

where $\beta$ is chosen to be reasonably large to ensure that the pairwise affinity is greater than zero.

Recall that the factorization (Eq. 2.6) reveals that the goal of GM (Eq. 2.7) is similar to ICP (Eq. 2.10). In order to make GM more robust to geometric deformations, DGM aims to find the optimal correspondence $\mathbf{X}$ as well as the optimal transformation $\mathcal{T}$ such that the global consistency can be maximized:

$$\max_{\mathbf{X} \in \Pi, \mathcal{T} \in \Psi} J_{dgm}(\mathbf{X}, \mathcal{T}) = \operatorname{tr}\left(\mathbf{K}_p(\mathcal{T})^T \mathbf{X}\right) + \lambda \operatorname{tr}\left(\mathbf{K}_q(\mathcal{T})^T \mathbf{Y}\right) - \psi(\mathcal{T}), \tag{2.20}$$

where $\lambda \geq 0$ is used to balance between the importance of the node and edge consistency. Similar to ICP, $\psi(\mathcal{T})$ and $\Psi$ are used to constrain and penalize the transformation parameter. Eq. 2.20 extends ICP by adding the transformation. In particular, if $\lambda = 0$, solving DGM is equivalent to ICP. In other case when $\lambda > 0$ and $\mathcal{T}$ is known, solving DGM is identical to a GM problem.

### 2.7.2 Optimization

Due to the non-convex nature of the objective, we optimize $J_{dgm}$ by alternatively solving the correspondence ($\mathbf{X}$) and the transformation parameter ($\mathcal{T}$). The initialization is important for the performance of DGM. However, how to select a good initialization is beyond the scope of this chapter and we simply set the initial transformation as an identity one, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p}$.

Optimizing Eq. 2.20 consists of alternation between optimizing for the correspondence and the geometric transformation. Given the transformation $\mathcal{T}$, DGM is equivalent to a traditional

[4]In addition to the linear affinity, there are other choices for defining the similarity between nodes and edges. For instance, the edge similarity could be computed as a RBF kernel. Due to the non-linearity of RBF, we might not have a closed-form solution for computing the transformation. But we could adopt Fitzgibbon's method [72] to optimize the transformation with respect to a linear approximation of the cost function.

GM problem. To find the node correspondence $\mathbf{X}$, we adopt the path-following algorithm by optimizing Eq. 2.15.

Given the correspondence matrix $\mathbf{X}$, the optimization over the transformation parameter $\mathcal{T}$ is similar to ICP. The main difficulty lies in the fact that the transformation parameter $\mathcal{T}$ appears not only in the node affinity $\mathbf{K}_p(\mathcal{T})$, but also in the edge affinity $\mathbf{K}_q(\mathcal{T})$. After some linear algebra, however, it can be shown that for certain choices of transformations in 2-D (*e.g.*, similarity, affine, RBF non-rigid), the parameter can be computed in closed-form.

**Similarity transformation:** According to the definition in Eq. 2.19, the similarity transformation of the edge feature is $\tau(\mathbf{q}) = s\mathbf{R}\mathbf{q}$ or $\tau(\mathbf{Q}) = s\mathbf{R}\mathbf{Q}$. Since the translation $\mathbf{t}$ only affects the node feature, the optimal $\mathbf{t}^*$ can be computed as follows once the optimal scalar $s^*$ and rotation $\mathbf{R}^*$ are known:

$$\mathbf{t}^* = \bar{\mathbf{p}}_1 - s^*\mathbf{R}^*\bar{\mathbf{p}}_2, \text{ where } \bar{\mathbf{p}}_1 = \frac{\mathbf{P}_1\mathbf{X}\mathbf{1}_{n_2}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}, \ \bar{\mathbf{p}}_2 = \frac{\mathbf{P}_2\mathbf{X}^T\mathbf{1}_{n_1}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}.$$

After centerizing the data, $\bar{\mathbf{P}}_1 = \mathbf{P}_1 - \bar{\mathbf{p}}_1\mathbf{1}_{n_1}^T$ and $\bar{\mathbf{P}}_2 = \mathbf{P}_2 - \bar{\mathbf{p}}_2\mathbf{1}_{n_2}^T$, it can be shown that the optimal rotation and scaling can be achieved at:

$$\begin{aligned}\mathbf{R}^* &= \mathbf{U}\,\mathrm{diag}(1, \cdots, |\mathbf{U}\mathbf{V}^T|)\mathbf{V}^T, \\ s^* &= \frac{\mathrm{tr}(\mathbf{\Sigma})}{\mathrm{tr}\left(\mathbf{1}_{n_1\times 2}(\bar{\mathbf{P}}_2 \circ \bar{\mathbf{P}}_2)\mathbf{X}^T\right) + \lambda_q\,\mathrm{tr}\left(\mathbf{1}_{m_1\times 2}(\mathbf{Q}_2 \circ \mathbf{Q}_2)\mathbf{Y}^T\right)},\end{aligned}$$

where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \bar{\mathbf{P}}_1\mathbf{X}\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_1\mathbf{Y}\mathbf{Q}_2^T$ is computed by SVD.

**Affine transformation:** After applying the transformation, the edge feature becomes $\tau(\mathbf{q}) = \mathbf{V}\mathbf{q}$ or $\tau(\mathbf{Q}) = \mathbf{V}\mathbf{Q}$. Similar to the similarity transformation, the optimal translation is dependent on the optimal affine matrix $\mathbf{V}^*$:

$$\mathbf{t}^* = \bar{\mathbf{p}}_1 - \mathbf{V}^*\bar{\mathbf{p}}_2, \text{ where } \bar{\mathbf{p}}_1 = \frac{\mathbf{P}_1\mathbf{X}\mathbf{1}_{n_2}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}, \ \bar{\mathbf{p}}_2 = \frac{\mathbf{P}_2\mathbf{X}^T\mathbf{1}_{n_1}}{\mathbf{1}_{n_1}^T\mathbf{X}\mathbf{1}_{n_2}}.$$

After centerizing all the data, we can compute the optimal affine transformation:

$$\begin{aligned}\mathbf{V}^* &= \mathbf{A}\mathbf{B}^{-1}, \\ \text{where } \mathbf{A} &= \bar{\mathbf{P}}_1\mathbf{X}\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_1\mathbf{Y}\mathbf{Q}_2^T, \ \mathbf{B} = \bar{\mathbf{P}}_2\,\mathrm{diag}(\mathbf{X}^T\mathbf{1}_{n_1})\bar{\mathbf{P}}_2^T + \lambda_q\mathbf{Q}_2\,\mathrm{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})\mathbf{Q}_2^T.\end{aligned}$$

**Non-rigid transformation:** According to the definition in Eq. 2.19, the non-rigid transformation of the edge feature is $\tau(\mathbf{q}_c) = \mathbf{q}_c + \mathbf{W}(\phi(\mathbf{p}_i) - \phi(\mathbf{p}_j))$, where edge $\mathbf{q}_c$ connects between

point $\mathbf{p}_i$ and $\mathbf{p}_j$. In matrix form, we can show that $\tau(\mathbf{Q}) = \mathbf{Q} + \mathbf{W}\mathbf{L}_q$, where $\mathbf{L}_q = \mathbf{L}_p(\mathbf{G} - \mathbf{H})$. Plugging the definition of $\tau(\mathbf{P})$ and $\tau(\mathbf{Q})$ in $J_{dgm}$ yields:

$$\begin{aligned} J_{dgm}(\mathbf{W}) = \operatorname{tr}\Big\{\Big(2\mathbf{P}_1^T\mathbf{W}\mathbf{L}_p - \mathbf{1}_{n_1\times 2}(2\mathbf{P}_2 \circ \mathbf{W}\mathbf{L}_p + \mathbf{W}\mathbf{L}_p \circ \mathbf{W}\mathbf{L}_p)\Big)\mathbf{X}^T\Big\} \\ + \lambda_q \operatorname{tr}\Big\{\Big(2\mathbf{Q}_1^T\mathbf{W}\mathbf{L}_q - \mathbf{1}_{m_1\times 2}(2\mathbf{Q}_2 \circ \mathbf{W}\mathbf{L}_q + \mathbf{W}\mathbf{L}_q \circ \mathbf{W}\mathbf{L}_q)\Big)\mathbf{Y}^T\Big\} - \lambda_w \operatorname{tr}(\mathbf{W}\mathbf{L}_p\mathbf{W}^T). \end{aligned}$$

Setting the gradient of the above objective to zero yields the optimal weight matrix $\mathbf{W}$ as:

$$\begin{aligned} \mathbf{W}^* &= \mathbf{A}\mathbf{B}^{-1}, \\ \text{where } \mathbf{A} &= \mathbf{P}_1\mathbf{X} - \mathbf{P}_2 \operatorname{diag}(\mathbf{X}^T\mathbf{1}_{n_1}) + \lambda_q\mathbf{Q}_1\mathbf{Y}(\mathbf{G}_2 - \mathbf{H}_2)^T - \lambda_q\mathbf{Q}_2 \operatorname{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})(\mathbf{G}_2 - \mathbf{H}_2)^T, \\ \mathbf{B} &= \mathbf{L}_p \operatorname{diag}(\mathbf{X}^T\mathbf{1}_{n_1}) + \lambda_w\mathbf{I}_{n_2} + \lambda_q\mathbf{L}_q \operatorname{diag}(\mathbf{Y}^T\mathbf{1}_{m_1})(\mathbf{G}_2 - \mathbf{H}_2)^T. \end{aligned}$$

### 2.7.3 Comparison with ICP

It is well known that the performance of ICP algorithms largely depends on the effectiveness of the initialization step. In the following example, we empirically illustrate how by adding additional pairwise constrains, DGM is less sensitive to the initialization. Fig. 2.8a illustrates the problem of aligning two fish shapes under varying values for the initial rotation and scale parameters. As shown in Fig. 2.8b, ICP gets trapped into a local optima if the orientation gap is larger than $\frac{1}{3}\pi$ (the error should be 0). Similarly, DGM fails for large orientation gap after two iterations (the left column of Fig. 2.8c). However, as the number of iterations increasing, DGM is able to match shapes with very large deformation in rotation and scales. After $24$ iterations, DGM ultimately finds the optimal matching for all the initializations (the right column of Fig. 2.8c). This experiment shows that adding pairwise constraints can make the ICP algorithm more robust to the problem of local optima.

## 2.8 Experiments

This section reports experimental results on four benchmark datasets and compares FGM to several state-of-the-art methods for GM and ICP. In the first three experiments, we test the performance of using the path-following algorithm for optimizing GM problems. In the last experiment

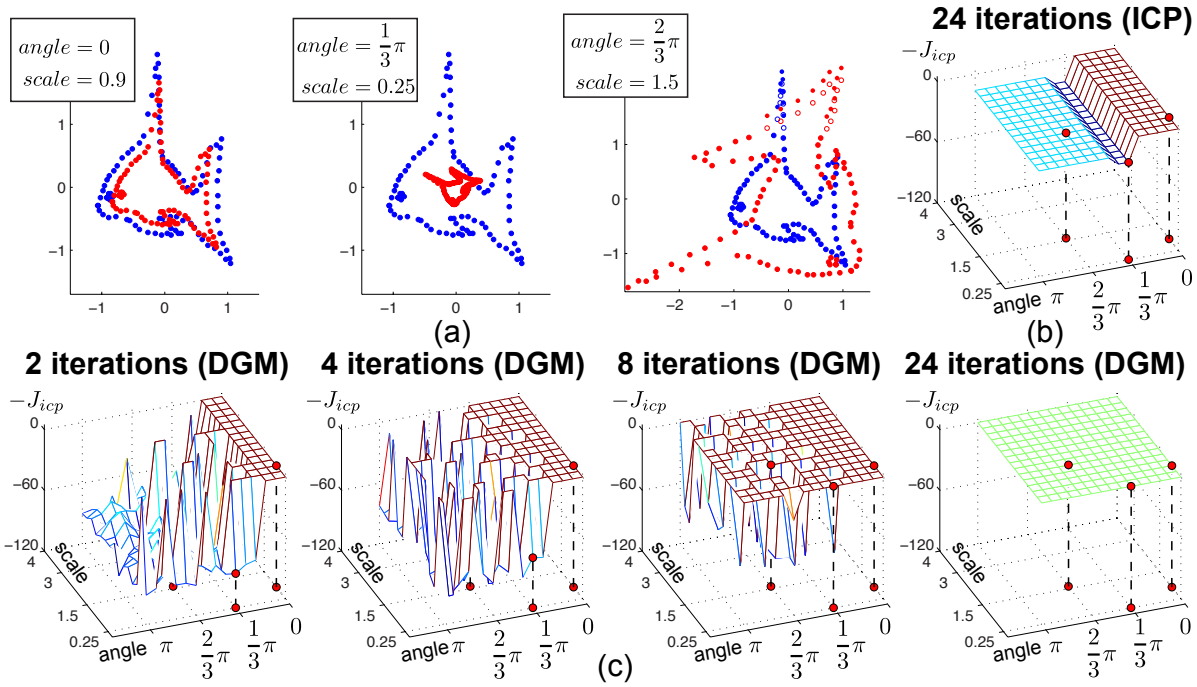


Figure 2.8: Comparison between ICP and DGM for aligning shapes for several initial values of rotation and scale parameters. (a) Examples of initializations for different angles and scales. (b) Objective surfaces obtained by ICP for different initializations. (c) Objective surfaces obtained by DGM.

we add a known geometrical transformation between graphs and compare with ICP algorithm on the problem of matching rigid and non-rigid shapes.

### 2.8.1 Baselines and evaluation metrics

In the experiment, we compared against seven state-of-the-art algorithms.

**Graduated assignment (GA):** GA [77] performs gradient ascent on a relaxation of Eq. 2.3 driven by an annealing schedule. At each step, it maximizes a Taylor expansion of the non-convex QP around the previous approximate solution. The accuracy of the approximation is controlled by a continuation parameter, $\beta_{t+1} \leftarrow \alpha\beta_t \leq \beta_{max}$. In all experiments, we set $\alpha = 1.075$, $\beta_0 = .5$ and $\beta_{max} = 200$.

**Spectral matching (SM):** SM [112] optimizes a relaxed problem of Eq. 2.3 that drops the

affine constraints and introduces a unit-length constraint on $\mathbf{X}$, that is:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \|\operatorname{vec}(\mathbf{X})\|_2^2 = 1.$$

The globally optimal solution of the relaxed problem is the leading eigenvector of $\mathbf{K}$.

**Spectral matching with affine constraints (SMAC):** SMAC [51] adds affine constraints to the SM problem maximizing:

$$\max_{\mathbf{X}} J_{gm}(\mathbf{X}), \quad \text{s.t. } \mathbf{A}\operatorname{vec}(\mathbf{X}) = \mathbf{b} \text{ and } \|\operatorname{vec}(\mathbf{X})\|_2^2 = 1.$$

The solution is also an eigenvalue problem.

**Integer projected fixed point method (IPFP):** IPFP [115] can take any continuous or discrete solution as inputs and iteratively improve the solution. In our experiments, we implemented two versions: (1) IPFP-U, which starts from the same initial $\mathbf{X}$ as our method; (2) IPFP-S, which is initialized by SM.

**Probabilistic matching (PM):** PM [216] designs the following convex objective function that can be globally optimized by applying the Sinkhorn's algorithm [176]:

$$\min_{\mathbf{X}\in\mathcal{D}} S(\mathbf{Y}\|\mathbf{X}),$$

where $S(\cdot)$ denotes the relative entropy error based on the Kullback-Leibler divergence and $\mathbf{Y} \in \mathbb{R}^{n_1\times n_2}$ is calculated by marginalizing $\mathbf{K}$.

**Re-weighted random walk matching (RRWM):** RRWM [43] introduces a random walk view on the problem and obtains the solution by simulating random walks with re-weighting jumps enforcing the matching constraints on the association graph. We fixed its parameters $\alpha = 0.2$ and $\beta = 30$ in all experiments.

**Concave optimization (CAV):** Concave optimization was firstly used by Maciel and Costeira [128] for optimizing GM. Compared to conventional iterative optimization method, concave optimization is guaranteed to converge at an integer solution and no further rounding step is needed. Similar to [128], we implemented CAV by employing the Frank-Wolfe algorithm to iteratively optimizing the concave relaxation (Eq. 2.14) of GM. To demonstrate the benefit of the path-following procedure, CAV was initialized by solving the same convex relaxation (Eq. 2.13) as FGM. The main difference between CAV and FGM is that the latter employs a path-following strategy to iteratively improve the result.

**Factorized graph matching (FGM):** We implemented two versions of our method, FGM-U for undirected graphs with only symmetric edge features and FGM-D for directed graphs with either symmetric or asymmetric edge features. FGM-U was first proposed in [222], which contains a similar factorization to Eq. 2.6. The main difference between FGM-U and FGM-D (this work) is that the former factorization can only be used for undirected graphs with symmetric edge features. In contrast, FGM-D can handle both undirected and directed graphs. To be able to use FGM-U for matching graphs with directed edges, we converted the directed edges to undirected edges. For each pair of undirected edges, we computed the new edge affinity as the average value of the original affinity between the directed edges. The parameters for our method were fixed to $\delta = 0.01$ in all experiments.

We used existing code from the author's websites for all methods. Notice that all methods except CAV, FGM-U and FGM-D need a post-processing step to discretize $\mathbf{X}$. To make a fair comparison, we applied the Hungarian algorithm to make this discretization. The code was implemented in Matlab on a laptop platform with 2.4G Intel Core 2 Duo and 4G memory. FGM-U and FGM-D were able to obtain the solution within a minute for graphs with $50$ nodes.

We evaluated both the matching accuracy and the objective score for the comparison of performance. The matching accuracy,

$$acc = \frac{\mathrm{tr}(\mathbf{X}_{alg}^T \mathbf{X}_{tru})}{\mathrm{tr}(\mathbf{1}_{n_2 \times n_1} \mathbf{X}_{tru})},$$

is calculated by computing the consistent matches between the correspondence matrix $\mathbf{X}_{alg}$ given by algorithm and ground-truth $\mathbf{X}_{tru}$. The objective score,

$$obj = \frac{J_{gm}(\mathbf{X}_{alg})}{J_{gm}(\mathbf{X}_{ours})},$$

is computed as the ratio between the objective values of our method (FGM-D) and other algorithms.

### 2.8.2 Synthetic dataset

This experiment performed a comparative evaluation of GM algorithms on randomly synthesized graphs following the experimental protocol of [43, 51, 77]. For each trial, we constructed two identical graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, each of which consists of $20$ inlier nodes and later we added $n_{out}$

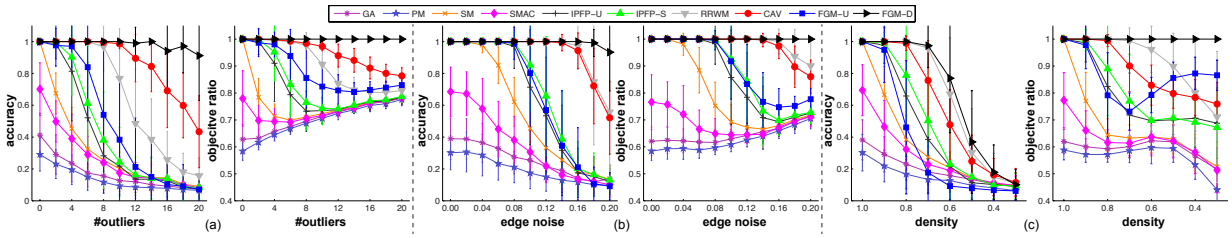


Figure 2.9: Comparison of GM methods on synthetic datasets. (a) Performance as a function of the outlier number ($n_{out}$). (b) Performance as a function of the edge noise ($\sigma$). (c) Performance as a function of the density of edges ($\rho$).

outlier nodes in both graphs. For each pair of nodes, the edge was randomly generated according to the edge density parameter $\rho \in [0, 1]$. Each edge in the first graph was assigned a random edge score distributed uniformly as $q_c^1 \sim \mathcal{U}(0, 1)$ and the corresponding edge $q_c^2 = q_c^1 + \epsilon$ in the second graph was perturbed by adding a random Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Notice that the edges had direction and the edge feature was asymmetrical. The edge-affinity matrix $\mathbf{K}_q$ was computed as $k^q_{c_1 c_2} = \exp(-\frac{(q^1_{c_1} - q^2_{c_2})^2}{0.15})$ and the node-affinity $\mathbf{K}_p$ was set to zero.

The experiment tested the performance of GM methods under three parameter settings. For each setting, we generated 100 different pairs of graphs and evaluated the average accuracy and objective ratio. In the first setting (Fig. 2.9a), we increased the number of outliers from $0$ to $20$ while fixing the noise $\sigma = 0$ and considering only fully connected graphs (*i.e.*, $\rho = 1$). In the second case (Fig. 2.9b), we perturbed the edge weights by changing the noise parameter $\sigma$ from $0$ to $0.2$, while fixing the other two parameters $n_{out} = 0$ and $\rho = 1$. In the last case (Fig. 2.9c), we verified the performance of matching sparse graphs by varying $\rho$ from $1$ to $0.3$. Under varying parameters, it can be observed that in most cases, our method, FGM-D, achieved the best performance over all other algorithms in terms of both accuracy and objective ratio. RRWM and CAV are our closest competitors. FGM-U did not achieve comparatively good results because it solved an undirected approximation of the original problem. Initialized by the same convex solution as FGM-D, however, CAV performed worse than FGM-D due to the lack of the robust path-following strategy.

### 2.8.3 House image dataset

The CMU house image sequence [1] was commonly used to test the performance of graph matching algorithms [37, 43, 60, 184]. This dataset consists of $111$ frames of a house, each of which has been manually labeled with $30$ landmarks. We used Delaunay triangulation to connect the landmarks. The edge weight $q_c$ was computed as the pairwise distance between the connected nodes. Due to the symmetry of distance-based feature, the edge was undirected. Given an image pair, the edge-affinity matrix $\mathbf{K}_q$ was computed by $k^q_{c_1c_2} = \exp(-\frac{(q^1_{c_1}-q^2_{c_2})^2}{2500})$ and the node-affinity $\mathbf{K}_p$ was set to zero. We tested the performance of all methods as a function of the separation between frames. We matched all possible image pairs, spaced exactly by $0:10:90$ frames and computed the average matching accuracy and objective ratio per sequence gap. Fig. 2.10a demonstrates an example pair of two frames.

We tested the performance of graph matching methods under two scenarios. In the first case (Fig. 2.10b) we used all $30$ nodes (*i.e.*, landmarks) and in the second one (Fig. 2.10c) we matched sub-graphs by randomly picking $25$ landmarks. It can be observed that in the first case (Fig. 2.10b), IPFP-S, RRWM, CAV, FGM-U and FGM-D almost obtained perfect matching of the original graphs. As some nodes became invisible and the graph got corrupted (Fig. 2.10c), the performance of all the methods degraded. However, FGM-U and FGM-D consistently achieved the best performance. The results demonstrate the advantages of the path-following algorithm over other state-of-the-art methods in solving general GM problems. In addition, it is interesting to notice that FGM-U and FGM-D performed similarly in both cases. This is because FGM-U can be considered as a special case of FGM-D when the graph consists of undirected edges.

### 2.8.4 Car and motorbike image dataset

The car and motorbike image dataset [2] was created in [116]. This dataset consists of $30$ pairs of car images and $20$ pairs of motorbike images taken from the PASCAL challenges. Each pair contains $30 \sim 60$ ground-truth correspondences. We computed for each node the feature, $p_i$, as the orientation of the normal vector to the contour. We adopted the Delaunay triangulation to build the graph. In this experiment, we considered the most general graph where the edge was

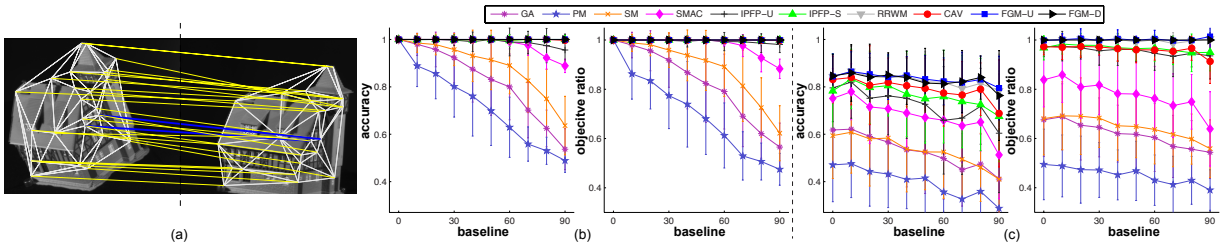


Figure 2.10: Comparison of GM methods on the CMU house datasets. (a) An example pair of frames with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) Performance of several algorithms using 30 nodes. (c) Performance using 25 nodes.

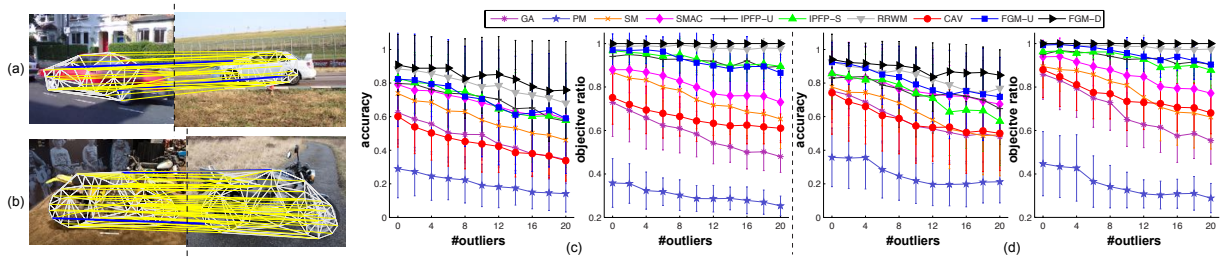


Figure 2.11: Comparison of GM methods for the car and motorbike dataset. (a) An example pair of car images with the correspondence generated by our method, where the blue lines indicate incorrect matches. (b) An example pair of motorbike images. (c) Performance as a function of the outlier number for the car images. (d) Performance as a function of the outlier number for the motorbike images.

directed and the edge feature was asymmetrical. More specifically, each edge was represented by a couple of values, $\mathbf{q}_c = [d_c, \theta_c]^T$, where $d_c$ was the pairwise distance between the connected nodes and $\theta_c$ was the angle between the edge and the horizontal line. Thus, for each pair of images, we computed the node affinity as $k^p_{ij} = \exp(-|p_i - p_j|)$ and the edge affinity as $k^q_{c_1c_2} = \exp(-\frac{1}{2}|d_{c_1} - d_{c_2}| - \frac{1}{2}|\theta_{c_1} - \theta_{c_2}|)$. Fig. 2.11a and Fig. 2.11b demonstrate example pairs of car and motorbike images respectively. To test the performance against noise, we randomly selected $0 \sim 20$ outlier nodes from the background. Similarly, we compared FGM-D against nine state-of-the-art methods. However, we were unable to directly use FGM-U to match directed graphs. Therefore, we ran FGM-U on an approximated undirected graph, where we computed the feature of the new undirected edge as the average value of the original affinities between the directed edges.

As observed in Fig. 2.11c-d, the proposed FGM-D consistently outperformed other methods on both datasets. Based on similar path-following strategy, however, FGM-U was unable to achieve the same accuracy because the graph consisted of directed edges and FGM-U was only applicable to undirected edges. It is worth to point that CAV performed not very well compared to FGM-D although it solved the same initial convex problem and optimized the same concave relaxation at the final step. This result as well as the previous experiment clearly demonstrated the path-following algorithm used by FGM-D provided a better optimization strategy than existing approaches. Finally, it is important to remind the reader that without the factorization proposed in this work it is not possible to apply the path-following method to general graphs.

### 2.8.5 Fish and character shape dataset

The UCF shape dataset [47] has been widely used for comparing ICP algorithms. In our experiment, we used two different templates. The first one has $91$ points sampled from the outer contour of a tropical fish. The second one consist of $105$ points sampled from a Chinese character. For each template, we designed two series of experiments to measure the robustness of an algorithm under different deformations and outliers. In the first series of experiments, we rotated the template with a varying degree (between $0$ and $\pi$). In the second set of experiments, a varying amount of outliers (between $0$ and $20$) were randomly added into the bounding box of

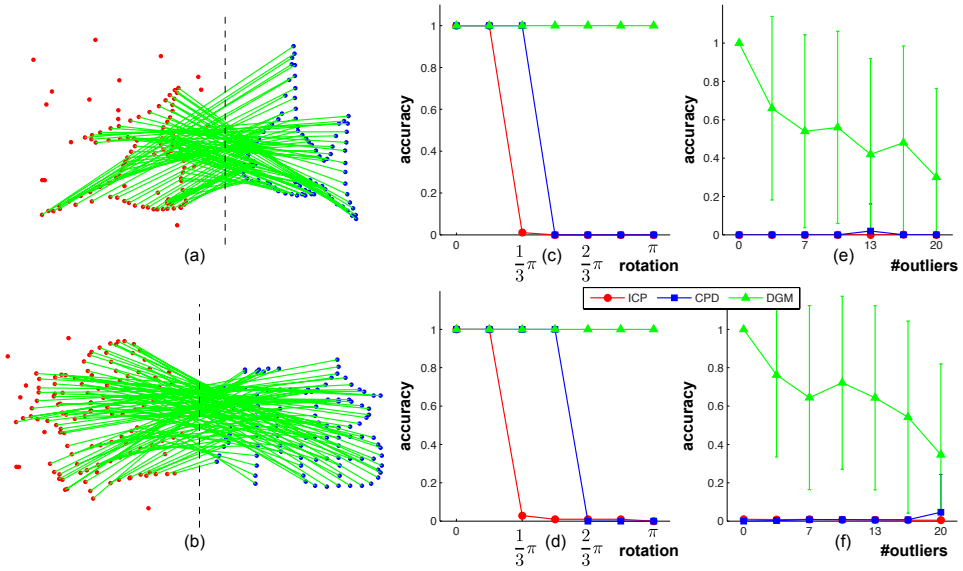


Figure 2.12: Comparison between DGM and ICP on the UCF shape datasets. (a-b) Two example pairs of shapes aligned using DGM. The red shape (left) is a rotated version of the blue one (right) by $\frac{2}{3}\pi$ and 20 random outliers were added. (c-d) Matching performance as a function of the initial rotations. (e-f) Matching performance as a function of the number of outliers.

template. For instance, Fig. 2.12a-b illustrate two pairs of example shapes with 20 outliers. We repeated the random generation 50 times for different levels of noise and compared DGM with the standard ICP algorithm and the coherent point drifting (CPD) [135]. The ICP algorithm was implemented by ourselves and CPD implementation was taken from the authors' website. We initialized all the algorithms with the same transformation, *i.e.*, $\tau(\mathbf{p}) = \mathbf{p}$. In DGM, Delaunay triangulation was employed to compute the graph structure. Recall that DGM simultaneously computes the correspondence and the rotation.

As shown in Fig. 2.12c-d, the proposed DGM can perfectly match the shapes across all the rotations without outliers, whereas both ICP and CPD get trapped in the local optimal when the rotation is larger than $\frac{2}{3}\pi$. When the number of outliers increases, DGM can still match most points under large rotation at $\frac{2}{3}\pi$. In contrast, ICP and CPD drastically failed in presence of outliers and large rotations (Fig. 2.12e-f).

In addition to a similarity transform, DGM can also incorporate non-rigid transformations in GM. Similar to the rigid case described in the main submission, we synthesized the non-rigid

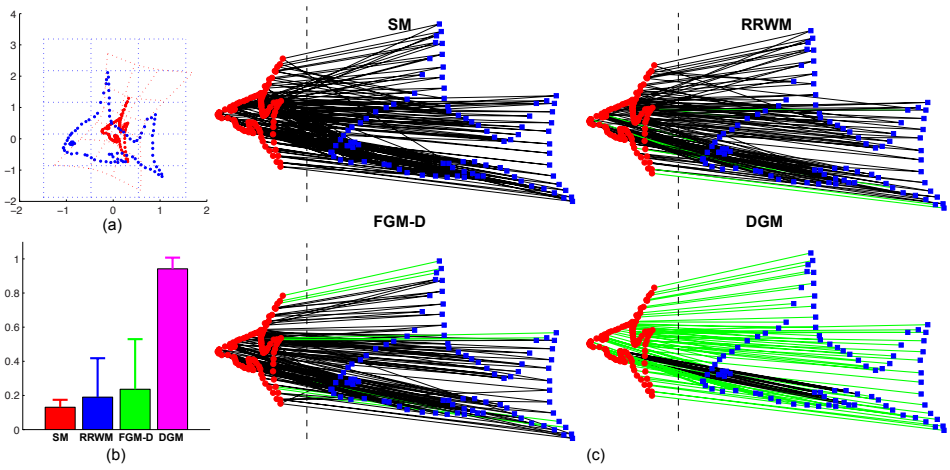


Figure 2.13: Comparison between DGM and GM methods for aligning non-rigidly deformed shapes. (a) An example of two fishes, where the red one is generated by a non-rigid transformation from the blue one. (b) Accuracy. (c) Results of GM methods, where the green and black lines indicate correct and incorrect correspondence respectively.

shape from the UCF shape dataset [47]. To generate the nonrigid transformation, we followed a similar setting in [135], where the domain of the point set was parameterized by a mesh of control points. The deformation of the mesh was modeled as an spline-based interpolation of the perturbation of the control points. We repeated the random generation $50$ times. For instance, Fig. 2.13a illustrates a synthetic pair of graphs.

We compared DGM with other two state-of-the-art GM methods: SM [112] and RRWM [43]. In addition, we tested the performance of our algorithm (FGM-D) only using the path-following algorithm for computing the correspondence but without estimating the transformation. As shown in Fig. 2.13b-c, FGM-D performed better than the other two GM methods. This is due to the path-following algorithm that is more accurate in optimizing GM problems. In addition, DGM significantly improved FGM-D by estimating the transformation.

## 2.9 Conclusions

This chapter proposes factorized graph matching (FGM), a new framework for better optimizing and understanding GM problems. The key idea of FGM is a novel factorization of the pairwise

affinity matrix into six smaller components. Four main benefits follow from factorizing the affinity matrix. First, there is no need to explicitly compute the costly affinity matrix. Second, it allows for a unification of GM methods as well as provides a clean connection with existing ICP algorithms. Third, the factorization enables the use of path-following algorithms that improve the performance of GM methods. Finally, it becomes easier to incorporate global geometrical transformation in GM.

In the chapter we have illustrated the advantages of factorizing the pair-wise affinity matrix of typical graph matching problems. The most computationally consuming part of the algorithm is the large number of iterations needed for FW method to converge when $J_\alpha$ is close to a convex function. Therefore, more advanced techniques (*e.g.*, conjugate gradient) can be used to speedup FW. In addition, we are currently exploring the extension of this factorization methods to other higher-order graph matching problems [41, 60, 216] as well as learning parameters for graph matching [37, 116].

# Chapter 3

# Temporal Matching

## 3.1 Introduction

Temporal alignment of multiple time series is an important problem with applications in many areas such as speech recognition [148], computer graphics [31], computer vision [40], and bioinformatics [3]. In particular, alignment of human motion from sensory data has recently received increasing attention in computer vision and computer graphics to solve problems such as curve matching [168], temporal clustering [227], tele-rehabilitation [205], activity recognition [97] and motion synthesis [88, 142]. While algorithms for aligning time series have been commonly used in computer vision and computer graphics, a relatively unexplored problem has been the alignment of multi-dimensional and multi-modal time series that encode human motion. Fig. 3.1 illustrates the main problem addressed by this chapter: how can we efficiently find the temporal correspondence between (1) frames of a video, (2) samples of motion capture data and (3) samples of 3-axis accelerometers of different subjects performing a similar action?

The alignment of multi-dimensional, multi-modal time series of human motion poses several challenges. First, there is typically a large variation in the subjects' physical characteristics, motion style and speed performing the activity. Second, large changes in view point also complicate the correspondence problem [192]. Third, it is unclear how existing techniques can be used to align sets of time series that have different modalities (*e.g.*, video and motion capture data). While there is extensive literature on time series alignment [82], standard extensions of dynamic

time warping (DTW) or Bayesian networks are not able to align multi-modal data.

To address these problems, this chapter proposes generalized canonical time warping (GCTW), a technique to temporally align multi-modal time series of different subjects performing similar activities. GCTW is a spatio-temporal alignment method that temporally aligns two or more multi-dimensional and multi-modal time series by maximizing the correlation across them. GCTW can be seen as an extension of DTW or canonical correlation analysis (CCA). To accommodate for subject variability and take into account the difference in the dimensionality of the signals, GCTW uses CCA as a measure of spatial correlation. GCTW extends DTW by incorporating a feature weighting mechanism to provide greater importance to more highly correlated features and align signals of different dimensionality. It also extends DTW by parameterizing the warping path as combination of monotonic functions, providing more accurate alignment and faster optimization strategies. Unlike exact approaches based on DTW, which have quadratic cost, GCTW uses a Gauss-Newton algorithm that has linear complexity in the length of the sequence. Preliminary versions of this chapter were published in [221, 223].

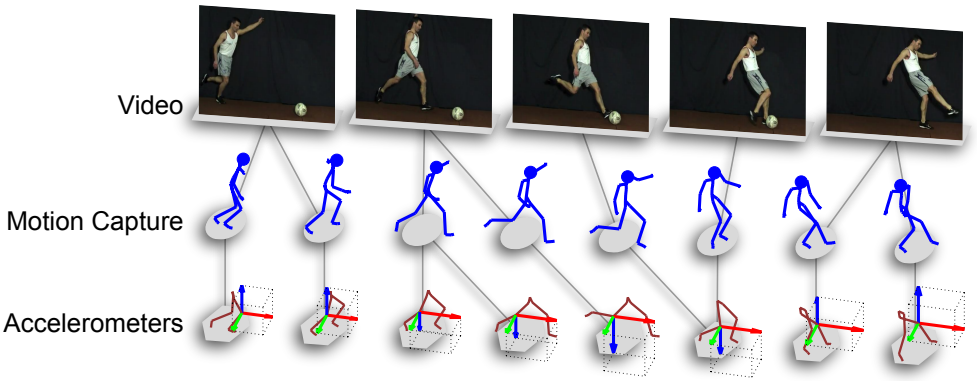


Figure 3.1: Temporal alignment of sequences recorded with different sensors of three subjects kicking a ball.

## 3.2 Previous work

### 3.2.1 Temporal alignment

This section discusses prior work on alignment of time series in the context of computer graphics, computer vision and data mining.

In the literature of computer graphics, temporal alignment of human motion has been commonly applied to solve problems such as content modeling [26], and motion blending [31]. Hsu *et al.*[88] proposed iterative motion warping, a robust method that finds a spatio-temporal warping between two instances of motion captured data. Shapiro *et al.*[169] used independent component analysis to separate motion data into visually meaningful components called style components. In comparison, our method solves a more general problem of aligning human motion from multi-modal time series.

In the context of computer vision, temporal alignment of video captured with different cameras and view points has been a topic of interest. Rao *et al.* [153] aligned trajectories of two moving points using constraints from the fundamental matrix. Junejo *et al.*[97] adopted DTW for synchronizing human actions with view changes based on a view-invariant descriptor. Wang and Wu [199] proposed a discriminative method to align two action sequences and measure their matching score. In comparison, our method simultaneously estimates the optimal spatial transformation and temporal correspondence to align video sequences. Recently, Gong and Medioni [78] extended our CTW approach to incorporate more complex spatial transformations through manifold learning. Nicolaou *et al.* [138] proposed a probabilistic extension of CTW for fusing multiple continuous expert annotations in tasks related to affective behavior. These works show the flexibility of our framework for aligning various types of sequential data.

In the field of data mining, there have been several extensions of DTW to align time series that differ in the temporal and spatial domain. Keogh and Pazzani [99], for example, used derivatives of the original signal to improve alignment with DTW. Listgarten *et al.*[119] proposed continuous profile models, a probabilistic method for simultaneously aligning and normalizing sets of time series in bio-informatics. Unlike these works, which were originally designed for aligning 1-D time series, our work addresses the more challenging problem of aligning multi-modal and

multi-dimensional time series.

In the literature of manifold alignment, Ham *et al.* [83] aligned manifolds of images in a semi-supervised manner. The prior knowledge of pairwise correspondences between two sets was used to guide the graph embedding. Wang and Mahadevan [196] aligned manifolds based on an extension of the Procrustes Analysis (PA). A main benefit of this approach is that PA learns a mapping generalized for out-of-sample cases. However, these models lack a mechanism to enforce temporal continuity.

### 3.2.2 Canonical correlation analysis (CCA)

CCA [10] is a technique to extract common features from a pair of multi-variate data. Given two sets of $n$ variables, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d_x \times n}$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n] \in \mathbb{R}^{d_y \times n}$, CCA finds the linear combinations of the variables in $\mathbf{X}$ that are most correlated with the linear combinations of the variables in $\mathbf{Y}$. Assuming zero-mean data ($\sum_i \mathbf{x}_i = \sum_j \mathbf{y}_j = 0$), CCA finds a combination of the original features that minimizes the sum of the distances between samples:

$$\min_{\{\mathbf{V}_x, \mathbf{V}_y\} \in \Phi} \; J_{cca} = \|\mathbf{V}_x^T \mathbf{X} - \mathbf{V}_y^T \mathbf{Y}\|_F^2 + \phi(\mathbf{V}_x) + \phi(\mathbf{V}_y), \tag{3.1}$$

where $\mathbf{V}_x \in \mathbb{R}^{d_x \times d}$ and $\mathbf{V}_y \in \mathbb{R}^{d_y \times d}$ denote the low-dimensional embeddings ($d \leq \min(d_x, d_y)$) for $\mathbf{X}$ and $\mathbf{Y}$ respectively. $\phi(\cdot)$ is a regularization function that penalizes the high-frequency of the embedding matrices:

$$\phi(\mathbf{V}) = \frac{\lambda}{1-\lambda} \|\mathbf{V}\|_F^2, \tag{3.2}$$

In order to avoid the trivial solution of $\mathbf{V}_x^T \mathbf{X}$ and $\mathbf{V}_y^T \mathbf{Y}$ being zero, CCA decorrelates the canonical variates (columns of $\mathbf{V}_x^T \mathbf{X}$ and $\mathbf{V}_y^T \mathbf{Y}$) by imposing the orthogonal constraint as:

$$\Phi = \Big\{ \{\mathbf{V}_x, \mathbf{V}_y\} \;\Big|\; \mathbf{V}_x^T \Big( (1-\lambda)\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I} \Big) \mathbf{V}_x = \mathbf{I}, \mathbf{V}_y^T \Big( (1-\lambda)\mathbf{Y}\mathbf{Y}^T + \lambda \mathbf{I} \Big) \mathbf{V}_y = \mathbf{I} \Big\}. \tag{3.3}$$

where $\lambda \in [0, 1]$ is a weight to trade-off between the least-square error and the regularization terms. In the case of insufficient samples where the covariance matrices ($\mathbf{X}\mathbf{X}^T$ or $\mathbf{Y}\mathbf{Y}^T$) are singular, a positive regularization term $\lambda \mathbf{I}$ is necessary to avoid over-fitting and numerical stability.

Optimizing Eq. 3.1 has a closed-form solution in terms of a generalized eigenvalue problem, *i.e.*, $[\mathbf{V}_x; \mathbf{V}_y] = \text{eig}_d(\mathbf{A}, \mathbf{B})$, where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{X}\mathbf{Y}^T \\ \mathbf{Y}\mathbf{X}^T & \mathbf{0} \end{bmatrix}, \mathbf{B} = (1-\lambda) \begin{bmatrix} \mathbf{X}\mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}\mathbf{Y}^T \end{bmatrix} + \lambda \mathbf{I}.$$

See [53] for a unification of several component analysis methods and a review of numerical techniques to efficiently solve generalized eigenvalue problems.

In computer vision, CCA has been used for matching sets of images in problems such as activity recognition from video [101] and activity correlation from cameras [125]. Recently, Fisher *et al.*[70] proposed an extension of CCA with parameterized warping functions to align protein expressions. The learned warping function is a linear combination of hyperbolic tangent functions with non-negative coefficients, ensuring monotonicity. Similarly, Shariat and Pavlovic [171] imposed monotonic constraints on CCA using non-negative least squares for activity recognition tasks. However, these methods were unable to deal with feature weighting.

### 3.2.3 Dynamic time warping (DTW)

Given two time series, $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_{n_x}] \in \mathbb{R}^{d \times n_x}$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_{n_y}] \in \mathbb{R}^{d \times n_y}$, DTW [148] aligns $\mathbf{X}$ and $\mathbf{Y}$ such that the sum of the distances between the aligned samples is minimized:

$$\min_{\{\mathbf{p}_x, \mathbf{p}_y\} \in \Psi} J_{dtw} = \sum_{t=1}^{l} \|\mathbf{x}_{p_t^x} - \mathbf{y}_{p_t^y}\|_2^2, \tag{3.4}$$

where $l \geq \max(n_x, n_y)$ is the number of indices used to align the samples. The optimal $l$ is automatically selected by the DTW algorithm. The warping paths, $\mathbf{p}_x \in \{1 : n_x\}^l$ and $\mathbf{p}_y \in \{1 : n_y\}^l$, denote the composition of alignment in frames. The $i^{th}$ frame in $\mathbf{X}$ and the $j^{th}$ frame in $\mathbf{Y}$ are aligned if there exists $p_t^x = i$ and $p_t^y = j$ at some step $t$.

In order to find a polynomial time solution, the warping paths must satisfy three constraints:

$$\Psi = \Big\{ \{\mathbf{p}_x, \mathbf{p}_y\} \;\Big|\; \mathbf{p}_x \in \{1 : n_x\}^l \text{ and } \mathbf{p}_y \in \{1 : n_y\}^l, \tag{3.5}$$

$$\textbf{Boundary: } [p_1^x, p_1^y] = [1, 1] \text{ and } [p_l^x, p_l^y] = [n_x, n_y],$$

$$\textbf{Monotonicity: } t_1 \geq t_2 \Rightarrow p_{t_1}^x \geq p_{t_2}^x \text{ and } p_{t_1}^y \geq p_{t_2}^y,$$

$$\textbf{Continuity: } [p_t^x, p_t^y] - [p_{t-1}^x, p_{t-1}^y] \in \{[0, 1], [1, 0], [1, 1]\} \Big\}.$$

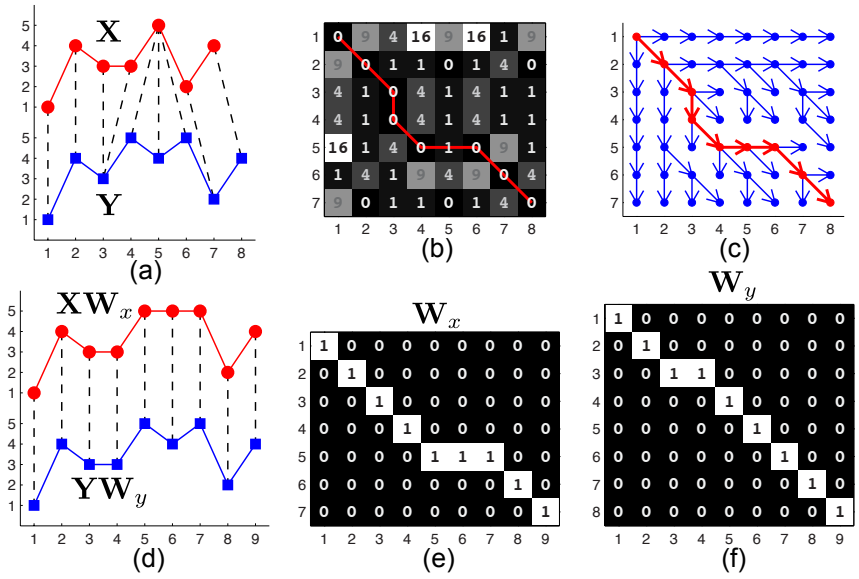


Figure 3.2: An example of DTW for aligning time series. (a) Two 1-D time series ($n_x = 7$ and $n_y = 8$) and the optimal alignment between samples computed by DTW. (b) Euclidean distances between samples, where the red curve denotes the optimal warping path ($l = 9$). (c) DP policy at each pair of samples, where the three arrow directions, $\downarrow, \searrow, \rightarrow$, denote the policy, $\pi(\cdot,\cdot) \in \{[1,0],[1,1],[0,1]\}$, respectively. (d) A matrix-form interpretation of DTW as stretching the two time series in matrix products. (e) Warping matrix $\mathbf{W}_x$. (f) Warping matrix $\mathbf{W}_y$.

The choice of step size in the continuity constraint is not unique. For instance, replacing the step size by $\{[2,1],[1,2],[1,1]\}$ can avoid the degenerate case in which a single frame of one sequence is assigned to many consecutive frames in the other sequence. See [148] for an extensive review on several DTW's modifications to control the warping paths.

Although the number of possible ways to align $\mathbf{X}$ and $\mathbf{Y}$ is exponential in $n_x$ and $n_y$, dynamic programming (DP) [19] offers an efficient approach with complexity of $O(n_x n_y)$ to minimize $J_{dtw}$ using Bellman's equation:

$$J^*(p_t^x, p_t^y) = \min_{\pi(p_t^x, p_t^y)} \|\mathbf{x}_{p_t^x} - \mathbf{y}_{p_t^y}\|_2^2 + J^*(p_{t+1}^x, p_{t+1}^y),$$

where the cost-to-go value function, $J^*(p_t^x, p_t^y)$, represents the cost remaining starting at $t^{th}$ step using the optimum policy $\pi^*$. The policy function, $\pi(\cdot,\cdot) : \{1 : n_x\} \times \{1 : n_y\} \rightarrow \{[1,0],[0,1],[1,1]\}$, defines the deterministic transition between consecutive steps, $[p_{t+1}^x, p_{t+1}^y] =$

$[p_t^x, p_t^y] + \pi(p_t^x, p_t^y)$. Once the policy queue is known, the alignment steps can be recursively selected by backtracking, $p_l^x = n_x$ and $p_l^y = n_y$.

Fig. 3.2a shows an example of DTW for aligning two 1-D time series. Fig. 3.2b illustrates the Euclidean distance of each pair of samples. To compute the optimal warping path, DP efficiently enumerates all possible steps as in Fig. 3.2c from the upper-left corner to the bottom-right one. At the end, the optimal alignment (denoted as the red curve) can be computed by iteratively tracing back along the arrows.

Given two sequences of length $n_x$ and $n_y$, exact DTW has a computational cost in space and time of $O(n_x n_y)$. In practice, various modifications [148] on the step size, local weights and global constraints (*e.g.*, the Sakoe-Chiba and Itakura Parallelogram bands [148]) have been proposed to speed up the DTW computation as well as to better control the possible routes of the warping paths. In recent work [45, 86, 161], a multi-scale searching scheme has been shown to effectively generate a speedup from one to three orders of magnitude, compared to the classic DTW algorithm. More recently, Rakthanmanon *et al.* [149] have shown that DTW for mining 1-D sub-sequences can be scaled up to very large datasets using early-abandoning and cascading lower bounds. However, most of these works are originally designed for 1-D time series. In comparison, our method can be applied to deal with more general multi-dimensional sequences and align signals of different dimensionality.

## 3.3 Canonical time warping (CTW)

DTW lacks a feature weighting mechanism and thus it cannot be directly used for aligning multi-modal sequences (*e.g.*, video and motion capture) with different features. To address this issue, this section presents CTW, a unified framework that combines DTW with CCA.

### 3.3.1 Least-squares formulation for DTW

In order to have a compact and compressible energy function for CTW, it is important to note that the original objective of DTW (Eq. 3.4) can be reformulated in matrix form as

$$\min_{\{\mathbf{p}_x,\mathbf{p}_y\}\in\Psi} J_{dtw} = \|\mathbf{X}\mathbf{W}_x - \mathbf{Y}\mathbf{W}_y\|_F^2, \tag{3.6}$$

where $\mathbf{X} \in \mathbb{R}^{d\times n_x}$ and $\mathbf{Y} \in \mathbb{R}^{d\times n_y}$ denote the two time series to be aligned. $\mathbf{W}_x = \mathbf{W}(\mathbf{p}_x) \in \{0,1\}^{n_x\times l}$ and $\mathbf{W}_y = \mathbf{W}(\mathbf{p}_y) \in \{0,1\}^{n_y\times l}$ are two binary warping matrices (Fig. 3.2e-f) associated with the warping paths by a non-linear mapping,

$$\mathbf{W}(\mathbf{p}) : \{1:n\}^l \rightarrow \{0,1\}^{n\times l}, \tag{3.7}$$

which sets $w_{p_t,t} = 1$ for any step $t \in \{1 : l\}$ and zero otherwise. These warping matrices $\mathbf{W}_x$ and $\mathbf{W}_y$ can only replicate (possibly multiple times) samples of the original sequences $\mathbf{X}$ and $\mathbf{Y}$. Fig. 3.2d illustrates the fact that the DTW alignment in Fig. 3.2a can be equivalently interpreted as stretching the two time series $\mathbf{X}$ and $\mathbf{Y}$ by multiplying them with the warping matrices $\mathbf{W}_x$ and $\mathbf{W}_y$, respectively as shown in Fig. 3.2e-f. Note that Eq. 3.6 is very similar to CCA's objective (Eq. 3.1). CCA applies a linear transformation to combine the rows (features), while DTW applies binary transformations to replicate the columns (time).

### 3.3.2 Objective function of CTW

In order to accommodate for differences in style and subject variability, add a feature selection mechanism, and reduce the dimensionality of the signals, CTW adds a linear transformation ($\mathbf{V}_x \in \mathbb{R}^{d_x\times d}$ and $\mathbf{V}_y \in \mathbb{R}^{d_y\times d}$) as in CCA to the least-square form of DTW (Eq. 3.6). Moreover, this transformation allows alignment of temporal signals with different dimensionality (*e.g.*, video and motion capture). In a nutshell, CTW combines DTW and CCA by minimizing:

$$\min_{\{\mathbf{V}_x,\mathbf{V}_y\}\in\Phi,\{\mathbf{p}_x,\mathbf{p}_y\}\in\Psi} J_{ctw} = \|\mathbf{V}_x^T\mathbf{X}\mathbf{W}_x - \mathbf{V}_y^T\mathbf{Y}\mathbf{W}_y\|_F^2 + \phi(\mathbf{V}_x) + \phi(\mathbf{V}_y), \tag{3.8}$$

where $\mathbf{V}_x \in \mathbb{R}^{d_x\times d}$ and $\mathbf{V}_y \in \mathbb{R}^{d_y\times d}$ parameterize the spatial transformation and project the sequences into the same low-dimensional coordinate system. Constrained by Eq. 3.5, $\mathbf{W}_x$ and

$\mathbf{W}_y$ warp the signal in time to maximize the temporal correlation. Similar to CCA, $\phi(\cdot)$ is a regularization term (Eq. 3.2) for $\mathbf{V}_x$ and $\mathbf{V}_y$. In addition, the projections satisfy the constraints,

$$\Phi = \Big\{ \{\mathbf{V}_x, \mathbf{V}_y\} \;\Big|\; \mathbf{V}_x^T \Big( (1-\lambda)\mathbf{X}\mathbf{W}_x\mathbf{W}_x^T\mathbf{X}^T + \lambda\mathbf{I} \Big)\mathbf{V}_x = \mathbf{I}, \mathbf{V}_y^T \Big( (1-\lambda)\mathbf{Y}\mathbf{W}_y\mathbf{W}_y^T\mathbf{Y}^T + \lambda\mathbf{I} \Big)\mathbf{V}_y = \mathbf{I} \Big\},$$

where $\lambda \in [0, 1]$ is to trade-off between the least-square error and the regularization term.

Eq. 3.8 is the main contribution of this chapter. CTW is a clean extension of CCA and DTW to align two signals in space and time. It extends previous work on CCA by adding temporal alignment and on DTW by allowing a feature selection and dimensionality reduction mechanism for aligning signals of different dimensions.

### 3.3.3 Optimization of CTW

Optimizing $J_{ctw}$ is a non-convex optimization problem with respect to the warping matrices and projection matrices. We take a coordinate-descent approach that alternates between solving the temporal alignment using DTW, and computing the spatial projections using CCA.

Given the warping matrices, the optimal projection matrices are the leading $d$ generalized eigenvectors, *i.e.*, $[\mathbf{V}_x; \mathbf{V}_y] = \text{eig}_d(\mathbf{A}, \mathbf{B})$, where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{X}\mathbf{W}_x\mathbf{W}_y^T\mathbf{Y}^T \\ \mathbf{Y}\mathbf{W}_y\mathbf{W}_x^T\mathbf{X}^T & \mathbf{0} \end{bmatrix}, \mathbf{B} = (1-\lambda) \begin{bmatrix} \mathbf{X}\mathbf{W}_x\mathbf{W}_x^T\mathbf{X}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}\mathbf{W}_y\mathbf{W}_y^T\mathbf{Y}^T \end{bmatrix} + \lambda\mathbf{I}.$$

In most experiments, we initialized CTW by setting $\mathbf{W}_x$ and $\mathbf{W}_y$ to the warping that uniformly aligns the sequences, *i.e.*, all the non-zero values in $\mathbf{W}_x$ and $\mathbf{W}_y$ are on their diagonals, *i.e.*, the warping paths are computed as

$$\mathbf{p}_x = round(linspace(1, n_x, l))', \tag{3.9}$$

where $round(\cdot)$ and $linspace(\cdot)$ are MATLAB functions. But CTW can be initialized by any other time warping method such as DTW or iterative motion warping (IMW) [88]. The dimension $d$ can be selected to preserve a certain amount (*e.g.*, $90\%$) of the total correlation. Once the spatial transformation is computed, the temporal alignment is computed using standard approaches for DTW. Alternating between these two steps (spatial and temporal alignment) monotonically decreases $J_{ctw}$. $J_{ctw}$ is bounded below and the proposed algorithm will converge to a point.

## 3.4 Generalized canonical time warping (GCTW)

In the previous section, we described CTW to align two multi-modal sequences with different features. However, CTW has three main limitations inherited from DTW: (1) The exact computational complexity of DTW for multi-dimensional sequences is quadratic both in space and time; (2) CTW and its extensions address the problem of aligning two sequences, but it is unclear how to extend it to the alignment of multiple sequences; (3) The temporal alignment is computed using DTW, which relies on DP to find the optimal path. In some problems (*e.g.*, sub-sequence alignment) the warping path provided by DP is too rigid (*e.g.*, the first and the last samples have to match).

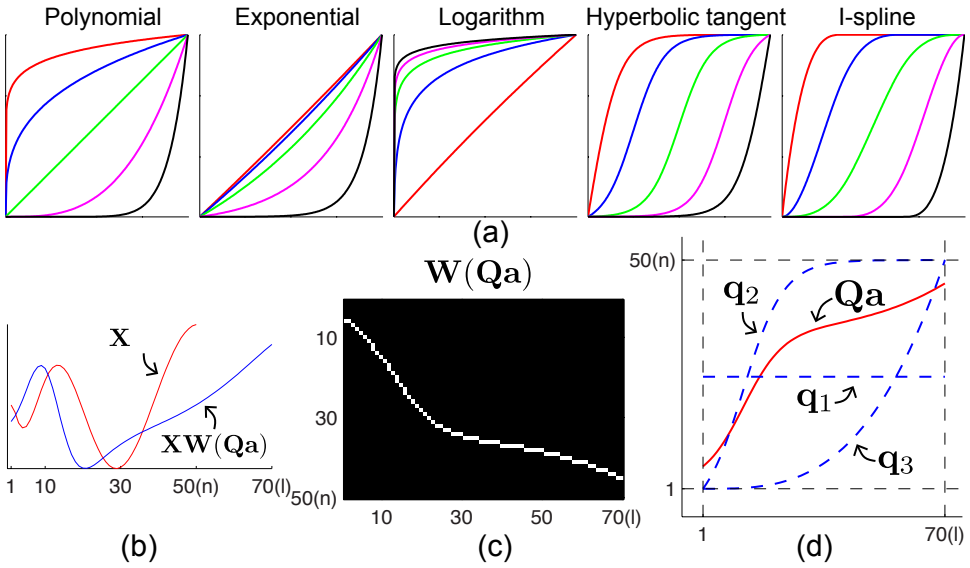


Figure 3.3: Approximating temporal warping using monotonic bases. (a) Five common choices for monotonic bases. (b) An example of time warping $\mathbf{XW}(\mathbf{Qa}) \in \mathbb{R}^{1\times 70}$ of 1-D time series $\mathbf{X} \in \mathbb{R}^{1\times 50}$. (c) The warping matrix. (d) The warping function $\mathbf{Qa}$ is a linear combination of three basis functions including a constant function ($\mathbf{q}_1$) and two monotonically functions ($\mathbf{q}_2$ and $\mathbf{q}_3$).

To address these issues, this section proposes GCTW, an efficient technique for spatio-temporal alignment of multiple time series. To accommodate for subject variability and to take into account the difference in the dimensionality of the signals, GCTW uses multi-set canonical

correlation analysis (mCCA). To compensate for temporal changes, GCTW extends DTW by incorporating a more efficient and flexible temporal warping parameterized by a set of monotonic basis functions. Unlike existing approaches based on DP with quadratic complexity, GCTW efficiently optimizes the time warping function using a Gauss-Newton algorithm, which has linear complexity.

### 3.4.1 Objective function of GCTW

Given a collection of $m$ time series, $\{\mathbf{X}_i\}_{i=1}^m$, GCTW aims to seek for each $\mathbf{X}_i = [\mathbf{x}_1^i, \cdots, \mathbf{x}_{n_i}^i] \in \mathbb{R}^{d_i \times n_i}$, a low-dimensional spatial embedding $\mathbf{V}_i \in \mathbb{R}^{d_i \times d}$ and a non-linear temporal transformation $\mathbf{W}_i = \mathbf{W}(\mathbf{p}_i) \in \{0,1\}^{n_i \times l}$ parameterized by $\mathbf{p}_i \in \{1 : n_i\}^l$, such that the resulting sequence $\mathbf{V}_i^T \mathbf{X}_i \mathbf{W}_i \in \mathbb{R}^{d \times l}$ is well aligned with the others in the least-squares sense. In a nutshell, GCTW minimizes the sum of pairwise distances:

$$\min_{\{\mathbf{V}_i\}_i \in \Phi, \{\mathbf{p}_i\}_i \in \Psi} J_{gctw} = \sum_{i=1}^{m} \sum_{j=1}^{m} \frac{1}{2} \|\mathbf{V}_i^T \mathbf{X}_i \mathbf{W}_i - \mathbf{V}_j^T \mathbf{X}_j \mathbf{W}_j\|_F^2 + \sum_{i=1}^{m} \Big( \phi(\mathbf{V}_i) + \psi(\mathbf{p}_i) \Big), \tag{3.10}$$

where $\phi(\mathbf{V}_i) = m\lambda/(1-\lambda)\|\mathbf{V}_i\|_F^2$ is the regularization function penalizing the irregularity of the spatial transformation $\mathbf{V}_i$. $\lambda \in [0,1]$ is a trade-off parameter between the least-square error and the regularization term. Following the multi-set canonical correlation analysis (mCCA) [84], GCTW constrains the spatial embeddings as:

$$\Phi = \Big\{ \{\mathbf{V}_i\}_i \;\Big|\; \sum_{i=1}^{m} \mathbf{V}_i^T \Big( (1-\lambda) \mathbf{X}_i \mathbf{W}_i \mathbf{W}_i^T \mathbf{X}_i^T + \lambda \mathbf{I} \Big) \mathbf{V}_i = \mathbf{I} \Big\}.$$

$\psi(\cdot)$ and $\Psi(\cdot)$, defined in the following sections, are used to respectively regularize and constrain the temporal transformation $\mathbf{p}_i$.

To be concise in the notation, let us consider a single sequence $\mathbf{X} \in \mathbb{R}^{d \times n}$ and its temporal warping, $\mathbf{p} \in \{1 : n\}^l$. While the possible composition of the temporal warping path, $\mathbf{p}$, is locally enforced by the original DTW constraints (Eq. 3.5), the global shape of any valid $\mathbf{p}$ must correspond to a monotonic and continuous trajectory in matrix $\mathbf{W} \in \{0,1\}^{n \times l}$ starting from the upper-left corner and ending at the bottom-right one. Recall that any positive combination of monotonic trajectories is guaranteed to be monotonic. GCTW parameterizes the warping path $\mathbf{p}$ as a linear combination of monotonic functions, that is:

$$\mathbf{p} \approx \sum_{c=1}^{k} a_c \mathbf{q}_c = \mathbf{Q}\mathbf{a}, \tag{3.11}$$

where $\mathbf{a} \in \mathbb{R}^k$ is the non-negative weight vector and $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_k] \in [1, n]^{l \times k}$ is the basis set composed of $k$ pre-defined monotonically increasing functions. Fig. 3.3a illustrates five common choices for $\mathbf{q}_c$, including (1) polynomial ($ax^b$), (2) exponential ($\exp(ax + b)$), (3) logarithm ($\log(ax + b)$), (4) hyperbolic tangent ($\tanh(ax + b)$) and (5) I-spline [151]. Similar work by Fisher *et al.* [70] also used hyperbolic tangent functions as temporal bases, and the weights were optimized using a non-negative least squares algorithm. However, GCTW differs in three aspects: (1) GCTW allows aligning multi-dimensional time series that have different features, while [70] can only align one-dimensional time-series; (2) GCTW uses a more efficient eigen decomposition to solve mCCA and quadratic programming for optimizing the weights; and (3) GCTW uses a family of monotonic functions that allow for a more general warping (*e.g.*, sub-sequence alignment), and constraints to regularize the solution.

To approximate the DTW constraints (Eq. 3.5) on the warping path $\mathbf{p}$, we alternatively impose the following constraints on the weights $\mathbf{a}$.

**Boundary conditions:** We enforce the position of the first frame, $p_1 = \mathbf{q}^{(1)}\mathbf{a} \geq 1$, and the last frame, $p_l = \mathbf{q}^{(l)}\mathbf{a} \leq n$, where $\mathbf{q}^{(1)} \in \mathbb{R}^{1 \times k}$ and $\mathbf{q}^{(l)} \in \mathbb{R}^{1 \times k}$ to be the first and last rows of the basis matrix $\mathbf{Q}$ respectively. In contrast to DTW, which imposes a tight boundary (*i.e.*, $p_1 = 1$ and $p_l = n$), GCTW allows $\mathbf{p}$ to index a sub-part of $\mathbf{X}$. This relaxation is useful in solving the more general problem of sub-sequence alignment. For instance, Fig. 3.4 illustrates an example of matching a shorter 1-D sequence (blue) to a sub-sequence of the longer one (red). In this sub-sequence alignment problem, GCTW models the time warping $\mathbf{p}$ as a combination of a linear basis $\mathbf{q}_1$ and a constant one $\mathbf{q}_2$.

**Monotonicity:** We enforce $t_1 \leq t_2 \Rightarrow p_{t_1} \leq p_{t_2}$ by constraining the sign of the weight: $\mathbf{a} \geq \mathbf{0}$. Note that constraining the weights is a sufficient condition to ensure monotonicity but it is not necessary. See [150, 156, 206] for in-depth discussions on monotonic functions.

**Continuity:** To approximate the hard constraint on the step size, GCTW penalizes the curvature of the warping path using a temporal regularization term, $\sum_{t=1}^{l} \|\nabla \mathbf{q}^{(t)}\mathbf{a}\|_2^2 \approx \|\mathbf{F}_l \mathbf{Q}\mathbf{a}\|_2^2$ where $\mathbf{F}_l \in \mathbb{R}^{l \times l}$ is the $1^{st}$ order differential operator.

In summary, we constrain the warping path in Eq. 3.10 adding the following constraints on

$\mathbf{a}$,

$$\psi(\mathbf{a}) = \eta\|\mathbf{F}_l\mathbf{Q}\mathbf{a}\|_2^2, \quad \Psi = \{\mathbf{a} \mid \mathbf{L}\mathbf{a} \leq \mathbf{b}\}, \text{ where } \mathbf{L} = [-\mathbf{I}_k; -\mathbf{q}^{(1)}; \mathbf{q}^{(l)}] \text{ and } \mathbf{b} = [\mathbf{0}_k; -1; n] \quad (3.12)$$

Therefore, given a basis set of $k$ monotone functions, all feasible weights belong to a polyhedron in $\mathbb{R}^k$ parameterized by $\mathbf{L} \in \mathbb{R}^{(k+2)\times k}$ and $\mathbf{b} \in \mathbb{R}^{k+2}$. For instance, Fig. 3.3d illustrates an example of a warping function (solid line) as a combination of three monotone functions (dotted lines).

### 3.4.2 Optimization of GCTW w.r.t. spatial basis

Minimizing $J_{gctw}$ (Eq. 3.10) is a non-convex optimization problem with respect to the temporal transformation and the spatial projection. We optimize GCTW by alternating between solving for the time warping using an efficient Gauss-Newton algorithm (discussed in the following section), and computing the spatial transformation using mCCA.

Assuming the time warping is fixed, mCCA computes the optimal $\{\mathbf{V}_i\}_i$ using a generalized eigen decomposition as $[\mathbf{V}_1; \cdots; \mathbf{V}_m] = \text{eig}_d(\mathbf{A}, \mathbf{B})$, where:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{X}_1\mathbf{W}_1\mathbf{W}_m^T\mathbf{X}_m^T \\ \vdots & \ddots & \vdots \\ \mathbf{X}_m\mathbf{W}_m\mathbf{W}_1^T\mathbf{X}_1^T & \cdots & \mathbf{0} \end{bmatrix}, \mathbf{B} = (1-\lambda)\begin{bmatrix} \mathbf{X}_1\mathbf{W}_1\mathbf{W}_1^T\mathbf{X}_1^T & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{X}_m\mathbf{W}_m\mathbf{W}_m^T\mathbf{X}_m^T \end{bmatrix} + \lambda\mathbf{I}.$$

These steps monotonically decrease $J_{gctw}$, and because the function is bounded below, the alternating scheme will converge to a critical point.

### 3.4.3 Optimization of GCTW w.r.t. temporal weights

By relaxing the warping path to be a linear combination of monotonic paths, Eq. 3.11 provides a new model for temporal alignment and new methods for optimizing it. Given $k$ basis functions, $\mathbf{Q} \in \mathbb{R}^{l\times k}$, optimizing Eq. 3.10 with respect to the warping paths $\{\mathbf{p}_i\}_i$ can be written as:

$$\min_{\{\mathbf{a}_i\}_i} J_a = \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{1}{2}\|\underbrace{\mathbf{V}_i^T\mathbf{X}_i\mathbf{W}(\mathbf{Q}\mathbf{a}_i)}_{\mathbf{Z}(\mathbf{a}_i)} - \underbrace{\mathbf{V}_j^T\mathbf{X}_j\mathbf{W}(\mathbf{Q}\mathbf{a}_j)}_{\mathbf{Z}(\mathbf{a}_j)}\|_F^2 + \sum_{i=1}^{m}\eta\|\mathbf{F}_l\mathbf{Q}\mathbf{a}_i\|_2^2, \text{s.t. } \mathbf{L}_i\mathbf{a}_i \leq \mathbf{b}_i, \ \forall i, \quad (3.13)$$

where $\mathbf{W}(\cdot)$ is a non-linear mapping function defined in Eq. 3.7. $\mathbf{L}_i$ and $\mathbf{b}_i$ are used to constrain the monotonicity and boundary of the time warping for each sequence. To shorten in notation, we denote each term $\mathbf{V}_i^T\mathbf{X}_i\mathbf{W}(\mathbf{Q}\mathbf{a}_i) \in \mathbb{R}^{d\times l}$ as $\mathbf{Z}(\mathbf{a}_i)$.

A direct optimization of $J_a$ is difficult due to the non-linear function $\mathbf{W}(\cdot)$. Inspired by the Lucas-Kanade framework for image alignment [126], we approximate the temporal alignment problem using a Gauss-Newton (GN) method. More specifically, GN iteratively updates the weights $\hat{\mathbf{a}}_i \leftarrow \mathbf{a}_i + \boldsymbol{\delta}_i$ by minimizing a series of first-order Taylor approximations of $J_a$ centered at each term $\mathbf{Z}(\mathbf{a}_i)$ given the initial $\mathbf{a}_i$, where $\boldsymbol{\delta}_i \in \mathbb{R}^k$ denotes the increment of the weight $\mathbf{a}_i$.

To better understand the approximation, let us first focus on, $\mathbf{z}_t(\mathbf{a}_i) \in \mathbb{R}^d$, the $t^{th}$ column of $\mathbf{Z}(\mathbf{a}_i) = [\mathbf{z}_1(\mathbf{a}_i), \cdots, \mathbf{z}_l(\mathbf{a}_i)]$, which can be rewritten as,

$$\mathbf{z}_t(\mathbf{a}_i) = [\mathbf{V}_i^T\mathbf{X}_i\mathbf{W}(\mathbf{Q}\mathbf{a}_i)]_t = \mathbf{V}_i^T\mathbf{X}_i[\mathbf{W}(\mathbf{Q}\mathbf{a}_i)]_t, \tag{3.14}$$

where $[\cdot]_t$ denotes the $t^{th}$ column of a matrix. According to the definition of $\mathbf{W}(\cdot)$ in Eq. 3.7, $[\mathbf{W}(\mathbf{Q}\mathbf{a}_i)]_t \in \{0,1\}^n$ is a binary vector with only one non-zero element located at $\mathbf{q}^{(t)}\mathbf{a}_i$, where $\mathbf{q}^{(t)} \in \mathbb{R}^{1\times k}$ is the $t^{th}$ row of $\mathbf{Q}$. In other words, $\mathbf{z}_t(\mathbf{a}_i)$ is a replication of $\mathbf{q}^{(t)}\mathbf{a}_i^{th}$ column of the signal $\mathbf{V}_i^T\mathbf{X}_i$, *i.e.*, $\mathbf{z}_t(\mathbf{a}_i) = [\mathbf{V}_i^T\mathbf{X}_i]_{\mathbf{q}^{(t)}\mathbf{a}_i}$. Following [126], we approximate $\mathbf{z}_t(\mathbf{a}_i + \boldsymbol{\delta}_i)$ as,

$$\mathbf{z}_t(\mathbf{a}_i + \boldsymbol{\delta}_i) \approx \mathbf{z}_t(\mathbf{a}_i) + \nabla(\mathbf{V}_i^T\mathbf{X}_i)|_{\mathbf{q}^{(t)}\mathbf{a}_i}\frac{\partial \mathbf{q}^{(t)}\mathbf{a}_i}{\partial \mathbf{a}_i}\boldsymbol{\delta}_i, \tag{3.15}$$

where $\nabla(\mathbf{V}_i^T\mathbf{X}_i)|_{\mathbf{q}^{(t)}\mathbf{a}_i} \in \mathbb{R}^d$ denotes the row-wise gradient[1] of the signal $\mathbf{V}_i^T\mathbf{X}_i$ around column $\mathbf{q}^{(t)}\mathbf{a}_i$. The term, $\partial\mathbf{q}^{(t)}\mathbf{a}_i/\partial\mathbf{a}_i = \mathbf{q}^{(t)} \in \mathbb{R}^{1\times k}$ is the Jacobian of the time warping.

Putting together the approximations of each column of $\mathbf{Z}(\mathbf{a}_i + \boldsymbol{\delta}_i)$ using Eq. 3.15 yields:

$$\mathrm{vec}\Big(\mathbf{Z}(\mathbf{a}_i + \boldsymbol{\delta}_i)\Big) \approx \mathbf{v}_i + \mathbf{G}_i\boldsymbol{\delta}_i, \text{where } \mathbf{v}_i = \mathrm{vec}\left(\mathbf{Z}(\mathbf{a}_i)\right), \mathbf{G}_i = \begin{bmatrix} \nabla(\mathbf{V}_i^T\mathbf{X}_i)|_{\mathbf{q}^{(1)}\mathbf{a}_i}\mathbf{q}^{(1)} \\ \vdots \\ \nabla(\mathbf{V}_i^T\mathbf{X}_i)|_{\mathbf{q}^{(l)}\mathbf{a}_i}\mathbf{q}^{(l)} \end{bmatrix}. \tag{3.16}$$

Plugging Eq. 3.16 in Eq. 3.13 yields:

$$J_a \approx \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{1}{2}\|\mathbf{v}_i + \mathbf{G}_i\boldsymbol{\delta}_i - \mathbf{v}_j - \mathbf{G}_j\boldsymbol{\delta}_j\|_2^2 + \sum_{i=1}^{m}\eta\|\mathbf{F}_i\mathbf{Q}(\mathbf{a}_i + \boldsymbol{\delta}_i)\|_2^2. \tag{3.17}$$

[1]The gradient is independently computed for each row of $\mathbf{V}_i^T\mathbf{X}_i$.

Minimizing Eq. 3.17 with respect to all the weight increments $\boldsymbol{\delta} = [\boldsymbol{\delta}_1; \cdots; \boldsymbol{\delta}_m] \in \mathbb{R}^{mk}$ yields a quadratic programming problem:

$$\min_{\boldsymbol{\delta}} \quad \frac{1}{2}\boldsymbol{\delta}^T \mathbf{H} \boldsymbol{\delta} + \mathbf{f}^T \boldsymbol{\delta}, \quad \text{s.t.} \quad \mathbf{L}\boldsymbol{\delta} \leq \mathbf{b} - \mathbf{L}\mathbf{a}, \tag{3.18}$$

whose components are computed as follows:

$$\mathbf{H} = \begin{bmatrix} m\mathbf{G}_1^T\mathbf{G}_1 + \eta\mathbf{Q}^T\mathbf{F}_1^T\mathbf{F}_1\mathbf{Q} & \cdots & -\mathbf{G}_1^T\mathbf{G}_m \\ \vdots & \ddots & \vdots \\ -\mathbf{G}_m^T\mathbf{G}_1 & \cdots & m\mathbf{G}_m^T\mathbf{G}_m + \eta\mathbf{Q}^T\mathbf{F}_m^T\mathbf{F}_m\mathbf{Q} \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} \mathbf{G}_1^T(m\mathbf{v}_1 - \sum_i \mathbf{v}_i) + \eta\mathbf{Q}^T\mathbf{F}_1^T\mathbf{F}_1\mathbf{Q}\mathbf{a}_1 \\ \vdots \\ \mathbf{G}_m^T(m\mathbf{v}_m - \sum_i \mathbf{v}_i) + \eta\mathbf{Q}^T\mathbf{F}_m^T\mathbf{F}_m\mathbf{Q}\mathbf{a}_m \end{bmatrix}, \mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_m \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_m \end{bmatrix}, \mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{L}_m \end{bmatrix}.$$

Note that the objective function of Eq. 3.18 is convex. Fig. 3.4 illustrates an example of aligning two 1-D time series (Fig. 3.4a) using this approach. To achieve sub-sequence alignment, we model the time warping path $\mathbf{p}$ as a combination of a linear basis $\mathbf{q}_1$ and a constant one $\mathbf{q}_2$ (Fig. 3.4d). As shown in Fig. 3.4b, Gauss-Newton takes three steps to find the optimal warping parameter in a 2-D space (Fig. 3.4c).

In most experiments, we initialized $\mathbf{a}_i$ by uniformly aligning the sequences (see GN-Init curve in Fig. 3.5b). However, better results can be achieved by using a more sophisticated method. The length of the warping path $l$ is usually set to be $l = 1.1 \max_{i=1}^{m} n_i$. The computational complexity of the algorithm is $O(dlmk + m^3k^3)$.

### 3.4.4 Comparison with other DTW techniques

As discussed in [148, 161], there are various techniques that have been proposed to accelerate and improve DTW. For instance, the Sakoe-Chiba band (DTW-SC) and the Itakura Parallelogram band (DTW-IP) reduce the complexity of the original DTW algorithm to $O(\beta n_x n_y)$ by constraining the warping path to be in a band of a certain shape, where $\beta < 1$ is the size ratio between the band and the original search space of DTW. However, using a narrow band (a small $\beta$) cuts off potential warping space, and may lead to a sub-optimal solution. For instance, Fig. 3.5a shows an example of two 1-D time series and the alignment results calculated by different algorithms. The results computed by DTW-SC and DTW-IP are less accurate than the ones computed using our

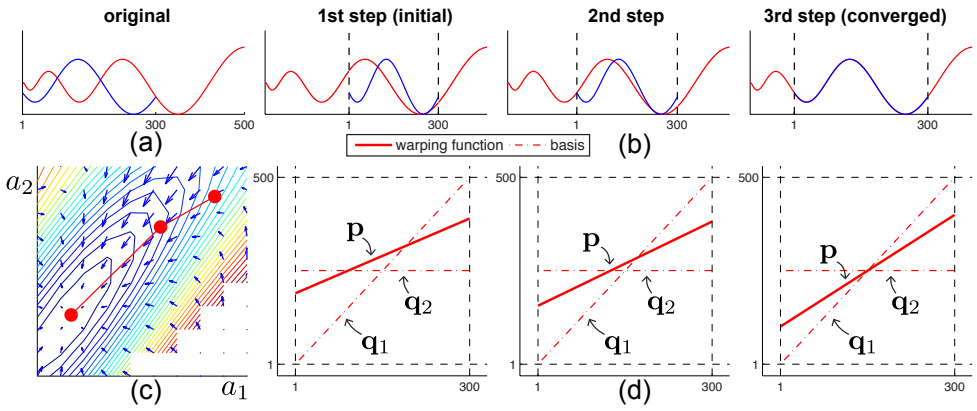


Figure 3.4: An example of using Gauss-Newton for solving the sub-sequence alignment problem. (a) Two 1-D sequences. (b) The Gauss-Newton optimization procedure, the longer red sequence is warped to match the shorter blue sequence. (c) The contour of the objective function ($J_a$ as defined in Eq. 3.13) with respect to the weights of two bases. (d) Warping function ($\mathbf{p}$) as a combination of a linear function ($\mathbf{q}_1$) and a constant function ($\mathbf{q}_2$) used for scaling and translation respectively.

proposed Gauss-Newton (GN). This is because both the SC and IP bands are over-constrained (Fig. 3.5b). Alternatively, instead of constraining the warping path, exhaustive DTW search can be approximated in a multi-level scheme. For instance, Salvador and Chan [161] introduced FastDTW by recursively projecting a solution from a coarser resolution and refining the projected solution in a higher resolution. Although the coarse-to-fine framework could largely reduce the search space, the solution is not exact. See Table. 3.1 for a detailed comparison.

To provide a quantitative evaluation, we synthetically generated 1-D sequences at $15$ scales. For DTW-SC, we set the bandwidth to be $\beta = 0.1$, which is common in practical applications. For FastDTW, we recursively shrink the sequence length in half from the finest level to the coarsest one. We then propagated the DTW solution from coarse to fine with radius $r = 5$. For GN, we varied $k$ to be $5, 8, 12$ to investigate the effect of the number of bases. For each scale, we randomly generated $100$ pairs of sequences. The error was computed using Eq. 3.20 and shown in Fig. 3.5c-d. DTW obtains the lowest error but takes the most time to compute. This is because DTW exhaustively searches the entire parameter space to find the global optima. DTW-SC, DTW-IP and FastDTW all need less time than DTW because they search a smaller space.

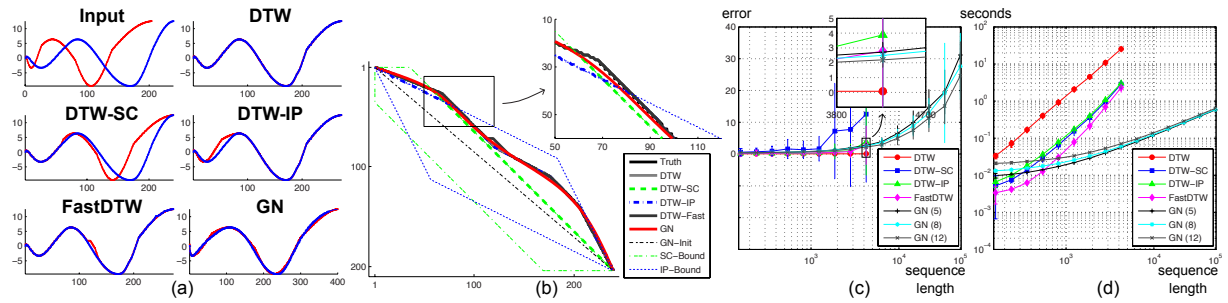


Figure 3.5: Comparison between Gauss-Newton and variants of DTW for temporal alignment. (a) An example of two 1-D time series and the alignment results calculated using the ground truth, DTW, DTW constrained in the Sakoe-Chiba band (DTW-SC), DTW constrained in the Itakura Parallelogram band (DTW-IP), DTW optimized in a multi-level scheme (FastDTW) and Gauss-Newton (GN). (b) Comparison of different warping paths. GN-Init denotes the initial warping used for GN. SC-Bound and IP-Bound denote the boundaries of SC band and IP band respectively. (c) Alignment errors. (d) Computational costs.

Empirically, DTW-SC is the least accurate compared to DTW-IP and FastDTW for our synthetic dataset. GN is most computationally efficient because it has linear complexity in the sequence length. Moreover, increasing the number of bases monotonically reduces the error.

## 3.5 Experiments

This section compares CTW and GCTW against state-of-the-art methods for temporal alignment of time series in seven experiments. In the first experiment, we compared the performance of CTW and GCTW against DTW, DDTW[99], and IMW [88] in the problem of aligning synthetic time series of varying complexity. In the second experiment, we aligned videos of different subjects performing a similar activity; each video is represented using different types of visual features. In the third experiment, we showed how GCTW and CTW can be applied to provide a metric useful for activity recognition. In the fourth experiment, we aligned facial expressions across subjects on videos with naturally occurring facial behavior. In the fifth experiment, we showed how GCTW can be applied to large-scale alignment. We aligned approximately $50,000$ frames of motion capture data of two subjects cooking the same recipe. In the sixth experiment,

we showed how GCTW can be used to localize common sub-sequences between two time series. The last experiment shows how GCTW is able to align three sequences of different subjects performing a similar action recorded with different sensors (motion capture data, accelerometers and video). In the first four experiments, the ground truth was known and we provided quantitative evaluation of the performance. In the others, we evaluated the quality of the alignment visually.

### 3.5.1 Evaluation methods

In the experiments, we compared CTW and GCTW with several state-of-the-art methods for temporal alignment of time series. Below, we provide a brief description of the techniques that we used for comparison.

**DTW and mDTW:** DTW is solved using a standard dynamic programming algorithm that minimizes Eq. 3.4. To evaluate the performance of temporal alignment of multiple sequences, we extended the concept of Procrustes analysis [59] to time series. That is, given $m$ $(> 2)$ time series, multi-sequence DTW (mDTW) seeks for a set of warping paths $\{\mathbf{p}_i\}_i$ that minimizes:

$$\min_{\{\mathbf{p}_i \in \Psi\}_i} J_{mdtw} = \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{1}{2}\|\mathbf{X}_i\mathbf{W}_i - \mathbf{X}_j\mathbf{W}_j\|_F^2 = m\sum_{i=1}^{m}\|\mathbf{X}_i\mathbf{W}_i - \frac{1}{m}\sum_{j=1}^{m}\mathbf{X}_j\mathbf{W}_j\|_F^2. \tag{3.19}$$

Based on the above fact, mDTW alternates between independently solving for each $\mathbf{p}_i$ using an asymmetrical DTW and updating the mean sequence $\frac{1}{m}\sum_{j=1}^{m}\mathbf{X}_j\mathbf{W}_j$ by averaging $\{\mathbf{X}_i\mathbf{W}_i\}_i$.

**DDTW and mDDTW:** In order to make DTW invariant to translation, derivative dynamic time warping (DDTW) [99] uses the derivatives of the original features and minimizes:

$$\min_{\{\mathbf{p}_x,\mathbf{p}_y\}\in\Psi} J_{ddtw} = \|\mathbf{X}\mathbf{F}_{n_x}^T\mathbf{W}_x - \mathbf{Y}\mathbf{F}_{n_y}^T\mathbf{W}_y\|_F^2,$$

where $\mathbf{F}_{n_x}$ and $\mathbf{F}_{n_y}$ are the $1^{st}$ order differential operators. To align multiple sequences, multi-sequence DDTW (mDDTW) extends DDTW in the Procrustes framework similar to Eq. 3.19.

**IMW and mIMW:** Similar to CTW, iterative motion warping (IMW) [88] alternates between time warping and spatial transformation to align two sequences. Assuming the same number of spatial features between $\mathbf{X} \in \mathbb{R}^{d\times n_x}$ and $\mathbf{Y} \in \mathbb{R}^{d\times n_y}$, IMW translates and re-scales each feature in $\mathbf{X}$ independently to match with $\mathbf{Y}$. Written in a simple matrix form, IMW minimizes:

$$\min_{\mathbf{p}_x\in\Psi,\mathbf{A}_x,\mathbf{B}_x} \quad J_{imw} = \|(\mathbf{X}\circ\mathbf{A}_x + \mathbf{B}_x)\mathbf{W}_x - \mathbf{Y}\|_F^2 + \lambda_a\|\mathbf{A}_x\mathbf{F}_{n_x}^T\|_F^2 + \lambda_b\|\mathbf{B}_x\mathbf{F}_{n_x}^T\|_F^2,$$

where $\mathbf{A}_x, \mathbf{B}_x \in \mathbb{R}^{d \times n_x}$ are the scaling and translation parameters respectively. $\lambda_a$ and $\lambda_b$ are the weights for the least-square error and the regularization terms. The regularization terms are used to enforce a smooth change in the columns of $\mathbf{A}_x$ and $\mathbf{B}_x$. In the experiments, we set them to be $\lambda_a = \lambda_b = 1$. IMW takes a coordinate-descent approach to optimize the time warping, scaling and translation. Given the warping matrix $\mathbf{W}_x$, the optimal spatial transformation can be computed in closed-form. To align multiple sequences, we extended IMW to multi-sequence IMW (mIMW) in the Procrustes framework similar to mDTW (Eq. 3.19).

**mCTW:** CTW was originally proposed to align two multi-modal sequences. We extended CTW to multi-sequence CTW (mCTW) for aligning multiple time series using the Procrustes analysis framework. mCTW optimizes the same objective (Eq. 3.10) as GCTW does. The main difference between mCTW and GCTW comes from the temporal alignment step. mCTW alternates between warping each time series using asymmetric DTW and updating the mean sequence, while GCTW uses Gauss-Newton for jointly optimizing over all weights of the bases.

Table. 3.1 compares temporal alignment methods in terms of the number of variables as well as the computational complexity. The comparison is divided into two cases, one for alignment of two sequences and another for alignment of more than two sequences. In the first case, given two time series, $\mathbf{X} \in \mathbb{R}^{d \times n_x}$ and $\mathbf{Y} \in \mathbb{R}^{d \times n_y}$, DTW and DDTW require the same complexity $O(n_x n_y)$ for finding the optimal $l$-length warping path. IMW additionally solves $d$ least-squares problems for each row of $\mathbf{A}_x$ and $\mathbf{B}_x$ independently. Similarly, CTW relies on DTW to optimize the time warping, resulting in a complexity of $O(n_x n_y)$ in both space and time. However, CTW uses CCA to accommodate the different in number of feature by solving a generalized eigen-decomposition of two $(d_x + d_y)$-by-$(d_x + d_y)$ matrices. CTW has fewer variables than IMW and thus is less likely to overfit the data. Compared to CTW, GCTW has the same complexity for computing the spatial embedding. The main advantage of GCTW is its Gauss-Newton component, which optimizes a small-scale QP with $2k$ variables for the time warping.

In the second case, given $m$ sequences, $\{\mathbf{X}_i \in \mathbb{R}^{d_i \times n_i}\}_{i=1}^m$, a direct generalization of the DTW is infeasible due to the combinatorial explosion of possible warpings, incurring a complexity of $O(\prod_{i=1}^m n_i)$. In the experiment, mDTW is used as an approximation of the exact DTW optimization. However, mDTW and other DTW-based methods (mDDTW, mIMW and mCTW) still have

quadratic complexity. Instead, GCTW approximates the combinatorial problem of time warping as a continuous optimization that can be more efficiently optimized by solving a small-scale QP with $mk$ variables.

### 3.5.2 Evaluation metrics

For all experiments, we used the normalized distance from the ground-truth as a metric for the error. More specifically, let us denote the alignment result of $m$ sequences by a set of time warping paths, $\mathbf{P}_{alg} = [\mathbf{p}_1^{alg}, \cdots, \mathbf{p}_m^{alg}] \in \mathbb{R}^{l_{alg} \times m}$, where $\mathbf{p}_i^{alg} \in \mathbb{R}^{l_{alg}}$ is the time warping path for the $i^{th}$ sequence. To evaluate the error of the time warping paths given by different methods, we computed their difference from the ground-truth path, $\mathbf{P}_{tru} = [\mathbf{p}_1^{tru}, \cdots, \mathbf{p}_m^{tru}] \in \mathbb{R}^{l_{tru} \times m}$, where the number of warping steps ($l_{alg}$ and $l_{tru}$) could be different. To better understand the error, let us consider each warping path $\mathbf{P} \in \mathbb{R}^{l \times m}$ as a curve in $\mathbb{R}^m$ with $l$ points (rows of $\mathbf{P}$). For instance, Fig. 3.7c and Fig. 3.9c compare the warping paths as 3-D and 2-D curves respectively. The error can be hence defined as the normalized distance between the curves $\mathbf{P}_{alg}$ and $\mathbf{P}_{tru}$,

$$error = \frac{1}{2}\Big(dist(\mathbf{P}_{alg}, \mathbf{P}_{tru}) + dist(\mathbf{P}_{tru}, \mathbf{P}_{alg})\Big), \text{ where } dist(\mathbf{P}_1, \mathbf{P}_2) = \frac{1}{l_1 l_2} \sum_{i=1}^{l_1} \min_{j=1}^{l_2} \|\mathbf{p}_1^{(i)} - \mathbf{p}_2^{(j)}\|_2. \tag{3.20}$$

The term, $\min_{j=1}^{l_2} \|\mathbf{p}_1^{(i)} - \mathbf{p}_2^{(j)}\|_2$, measures the shortest distance between the point $\mathbf{p}_1^{(i)}$ and any point on the curve $\mathbf{P}_2$, where $\mathbf{p}_1^{(i)} \in \mathbb{R}^{1 \times m}$ and $\mathbf{p}_2^{(j)} \in \mathbb{R}^{1 \times m}$ are the $i^{th}$ row of $\mathbf{P}_1$ and $j^{th}$ row of $\mathbf{P}_2$ respectively.

### 3.5.3 Aligning synthetic sequences

In the first experiment, we synthetically generated spatio-temporal signals (3-D in space and 1-D in time) to evaluate the performance of mCTW and GCTW. As shown in Fig. 3.6a, the signals were a randomly generated by spatially and temporally transforming a latent 2-D spiral, $\mathbf{Z} \in \mathbb{R}^{2 \times l}, l = 300$ as $\mathbf{X} = [(\mathbf{Z} + \mathbf{b}\mathbf{1}^T)\mathbf{M}; \mathbf{e}^T] \in \mathbb{R}^{3 \times n}$, where $\mathbf{U} \in \mathbb{R}^{2 \times 2}$ and $\mathbf{b} \in \mathbb{R}^2$ were a randomly generated projection matrix and translation vector respectively. To synthesize the temporal distortion, a binary selection matrix $\mathbf{M} \in \{0, 1\}^{l \times n}$ was generated by randomly choosing $n \leq l$ columns from the identity matrix $\mathbf{I}_l$. The third spatial dimension $\mathbf{e} \in \mathbb{R}^n$ was

| | Method | Degree-of-Freedom | | Complexity $O(\cdot)$ | |
|---|---|---|---|---|---|
| | | Embedding | Warping | Embedding | Warping |
| Two-sequence alignment $\mathbf{X} \in \mathbb{R}^{d_x \times n_x}$ $\mathbf{Y} \in \mathbb{R}^{d_y \times n_y}$ | DTW | – | $2l$ | – | $n_x n_y$ |
| | DTW-SC | – | $2l$ | – | $\beta n_x n_y$ |
| | DTW-IP | – | $2l$ | – | $\beta n_x n_y$ |
| | FastDTW | – | $2l$ | – | $r(n_x + n_y)/2$ |
| | DDTW | – | $2l$ | – | $n_x n_y$ |
| | IMW | $2dn_x$ | $2l$ | $dLS(2n_x)$ | $n_x n_y$ |
| | CTW | $d(d_x + d_y)$ | $2l$ | $eig(d_x + d_y)$ | $n_x n_y$ |
| | GCTW | $d(d_x + d_y)$ | $2k$ | $eig(d_x + d_y)$ | $QP(2k)$ |
| Multi-sequence alignment $\{\mathbf{X}_i \in \mathbb{R}^{d_i \times n_i}\}_i$ | DTW | – | $ml$ | – | $\Pi_i n_i$ |
| | mDTW | – | $ml$ | – | $l \sum_i n_i$ |
| | mDDTW | – | $ml$ | – | $l \sum_i n_i$ |
| | mIMW | $2l \sum_i d_i$ | $ml$ | $\sum_i d_i LS(2l)$ | $l \sum_i n_i$ |
| | mCTW | $d \sum_i d_i$ | $ml$ | $eig(\sum_i d_i)$ | $l \sum_i n_i$ |
| | GCTW | $d \sum_i d_i$ | $mk$ | $eig(\sum_i d_i)$ | $QP(mk)$ |

Table 3.1: Comparison of temporal alignment algorithms as a function of degrees-of-freedom and complexity. $l$ is the length of warping path. $LS(n)$, $QP(n)$ and $eig(n)$ denote the complexity of solving a least-squares of $n$ variables, a QP of $n$ variables and a generalized eigenvalue problem with two $n$-by-$n$ matrices, respectively.

added with a zero-mean Gaussian noise. In this experiment, mCTW and GCTW were compared with mDTW, mDDTW and mIMW for aligning three time series. The ground-truth alignment was known and the performance of each method was evaluated in terms of the alignment errors defined in Eq. 3.20. We repeated the above process $100$ times with random numbers. In each trial, we studied three different initialization methods for mCTW and GCTW: uniform alignment for mCTW-U and GCTW-U as in Eq. 3.9, mDTW for mCTW-D and GCTW-D, and mIMW for mCTW-I and GCTW-I. The subspace dimensionality for CCA $d$ was selected to preserve $90\%$ of the total correlation. In this case, we have sufficient samples ($l = 300$) in 3-D space, and the regularization weight $\lambda$ was set to zero. For GCTW, we selected three hyperbolic tangent and three polynomial functions as monotonic bases (upper-right corner in Fig. 3.6b).

Figs. 3.6 illustrates a comparison of the previously described methods for aligning multiple

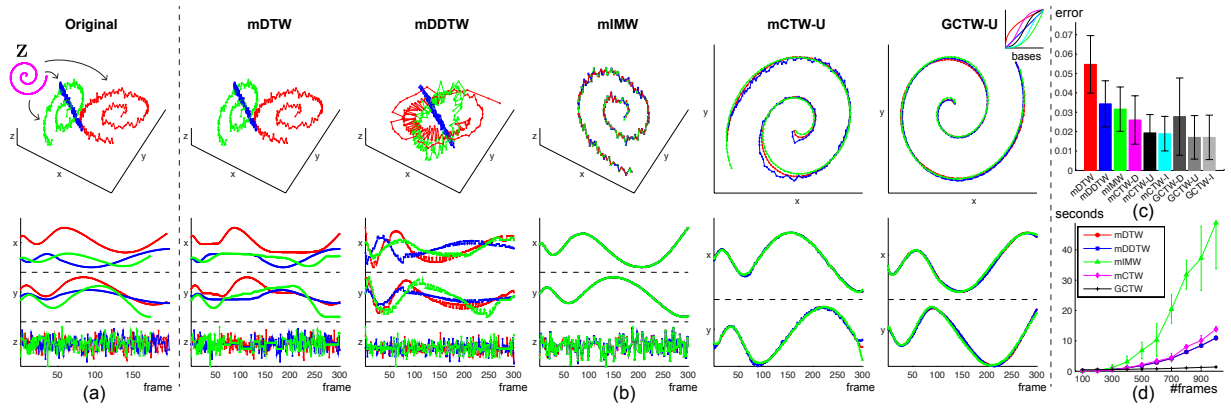


Figure 3.6: Comparison of temporal alignment algorithms on the synthetic dataset. (a) An example of three synthetic time series generated by performing a random spatio-temporal transformation of a 2-D latent sequence $\mathbf{Z}$ and adding Gaussian noise in the $3^{rd}$ dimension. (b) The alignment results. (c) Mean and variance of the alignment errors. (d) Mean and variance of the computational cost (time in seconds).

time series. Fig. 3.6b shows the spatio-temporal warping estimated by mDTW, mDDTW, mIMW, mCTW-U and GCTW-U. Fig. 3.6c shows the alignment errors (Eq. 3.20) for 100 randomly generated time series. Both mDTW and mDDTW performed poorly in this case since they do not have a feature weight mechanism to adapt the spatial transformation of the sequences. mIMW warps sequences towards others by translating and re-scaling each frame in each dimension. Moreover, mIMW has more parameters ($2l \sum_i d_i$) than mCTW and GCTW ($d \sum_i d_i$), and hence mIMW is more prone to over-fitting. Furthermore, mIMW tries to fit the noisy dimension ($3^{rd}$ spatial component), biasing alignment in time, whereas both mCTW and GCTW had a feature selection mechanism that effectively canceled out the third dimension. Among all the initializations, the ones initialized by uniform alignment (mCTW-U and GCTW-U) and mIMW (mCTW-I and GCTW-I) achieved the best results. However, mCTW and GCTW always improved the performance in comparison with the initializations. Compared to mCTW, GCTW achieved better performance when aligning more than two sequences because GCTW jointly optimizes over all the possible time warpings for each time series, while mCTW takes a greedy approach by warping each sequence towards the mean sequence independently.

Fig. 3.6d evaluates the computational cost of each method with respect to the average se-

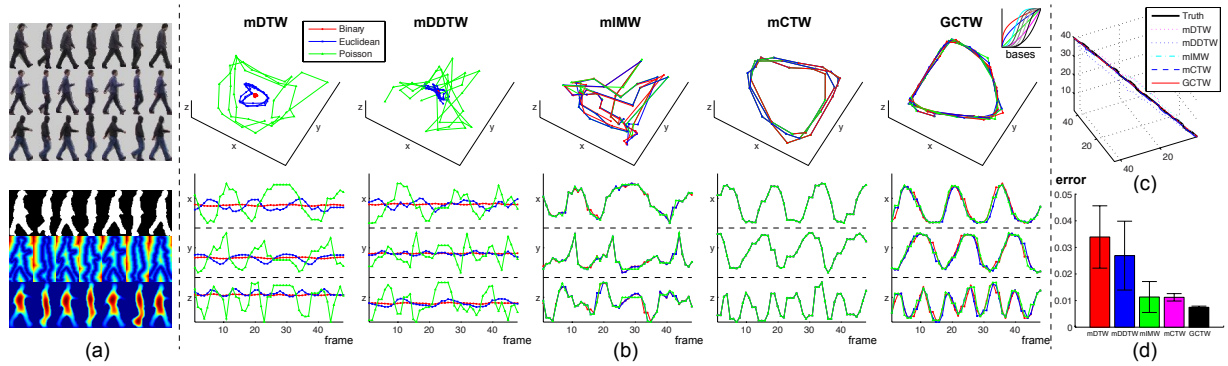


Figure 3.7: Comparison of temporal alignment algorithms for aligning multi-feature video data. (a) An example of three aligned videos by GCTW. The top three sequences are the original frames after background subtraction, while the bottom three are the binary images, the Euclidean distance transforms and the solutions of the Poisson equation. (b) The alignment results. (c) Comparison of time warping paths. (d) Mean and variance of the alignment errors.

quence length. mIMW was the most computationally intensive method because it solves a least-square problem for each feature dimension. mCTW was more expensive than DTW-based methods because of the additional computation to solve CCA. As expected, GCTW was the most efficient.

### 3.5.4 Aligning videos with different features

In the second experiment, we used mCTW and GCTW to align video sequences of different people performing a similar action. Each video was encoded using different visual features. The video sequences were taken from the Weizmann database [80], which contains nine people performing ten actions. To represent dynamic videos, we subtracted the background (the top three rows in Fig. 3.7a) and computed three popular shape features (the bottom three rows of Fig. 3.7a) for each 70-by-35 re-scaled mask image, including (1) binary image, (2) Euclidean distance transform [132], and (3) solution of Poisson equation [79]. In order to reduce the dimension of the feature space (2450), we picked the top 123 principal components that preserved 99% of the total energy. We randomly selected three sequences and manually labeled their temporal correspondence as ground-truth. We repeated the process 10 times. All methods were initial-

ized with uniform alignment. For GCTW, we used five hyperbolic tangent and five polynomial functions as the monotonic bases.

Fig. 3.7d shows the error for 10 randomly generated sets of videos. Neither mDTW nor mDDTW was able to align the videos because they were not able to handle alignment of signals of different dimensions. mIMW registered the top three components well in space; however, it overfitted the data and also computed a biased time warping path. In contrast, mCTW and GCTW warped the sequences accurately in both space and time.

### 3.5.5 Activity Classification

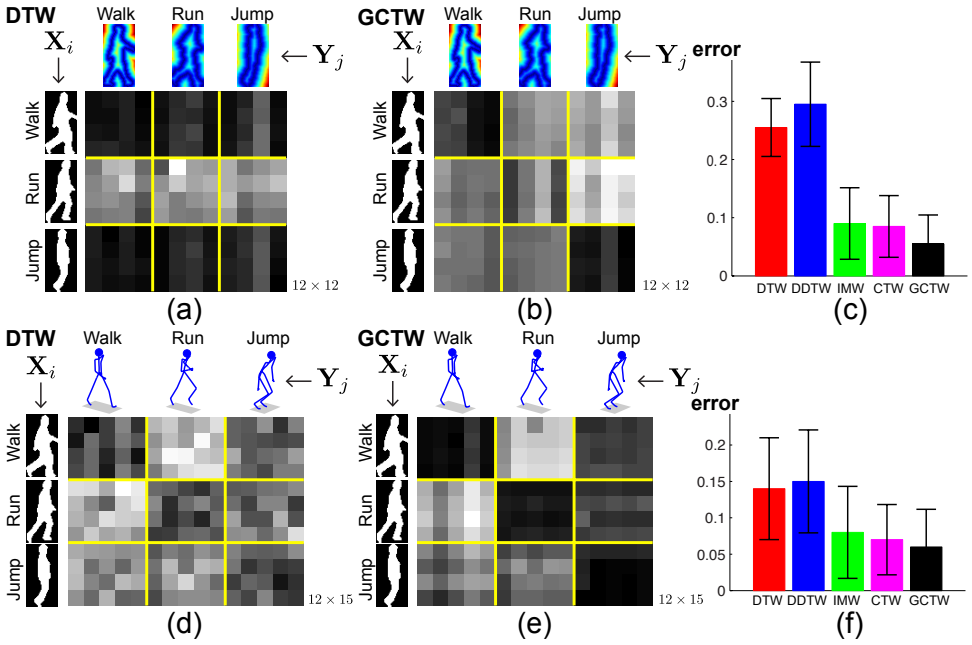


Figure 3.8: Activity classification. In (a-c), we computed the metric between videos using different features: binary images $\{\mathbf{X}_i\}_i$ and Euclidean distance transforms $\{\mathbf{Y}_j\}_j$. In (d-f), we computed the metric between video $\{\mathbf{X}_i\}_i$ and motion capture data $\{\mathbf{Y}_j\}_j$. (a)(d) DTW distance matrices. A darker color indicates a smaller distance. (b)(e) GCTW distance matrices. (c)(f) Mean and standard deviation of the classification error.

The previous section illustrated how to align multi-feature videos. This section explores the use of CTW and GCTW as distance metrics between time series. In particular, we evaluated

different methods for recognizing action in videos having different features, and between videos and motion capture data. The distance metrics were used in combination with a nearest neighbor classifier.

In our first experiment, given a training set of videos of a subject performing an activity recorded with different visual features (*e.g.*, binary silhouette, distance transform) $\{\mathbf{X}_i\}_i$, our goal is to find the testing video $\{\mathbf{Y}_j\}_j$ that contains similar activity. We took $24$ sequences from the Weizmann dataset [80]: $8$ people performed three actions (walking, running and jumping). We repeated our experiments $10$ times. In each trial, we randomly split the $24$ sequences into two disjoint sets of $12$ sequences used for training and testing respectively. The training sequences $\{\mathbf{X}_i\}_i$ were represented using the binary silhouette (see left columns of Fig. 3.8a-b as examples) while the testing ones $\{\mathbf{Y}_j\}_j$ were encoded with the Euclidean distance features (see top rows of Fig. 3.8a-b as examples). Given a pair of sequences, $\mathbf{X}_i \in \mathbb{R}^{d_x \times n_{x_i}}$ and $\mathbf{Y}_j \in \mathbb{R}^{d_y \times n_{y_j}}$, we computed the distance between videos using different methods: DTW, DDTW, IMW, CTW and GCTW. The warping path for each method is denoted as $\mathbf{W}_{x_i}^{alg} \in \mathbb{R}^{n_{x_i} \times l}$ and $\mathbf{W}_{y_j}^{alg} \in \mathbb{R}^{n_{y_j} \times l}$.

In order provide a fair comparison with DTW, DDTW and mIMW (that cannot compute distances between different features), we computed a pair of projection matrices, $\mathbf{V}_x \in \mathbb{R}^{d_x \times d}$ and $\mathbf{V}_y \in \mathbb{R}^{d_y \times d}$ for them. To do that, we took a subset of $1/3$ of the training sequences encoded with both features $\{\mathbf{X}_i^{tr}\}$ and $\{\mathbf{Y}_i^{tr}\}$, and uniformaly aligned each pair of sequences of the same label. The aligned frames were concatenated in two matrices $\mathbf{X}^{tr}$ and $\mathbf{Y}^{tr}$, and the projections $\mathbf{V}_x$ and $\mathbf{V}_y$ were computed by CCA optimizing Eq. 3.1. Recall that the projection matrices were fixed for DTW, DDTW and IMW during testing, but for CTW and GCTW they were optimized by the algorithms. Then, the distance between each pair of sequences was then computed as

$$dist(\mathbf{X}_i, \mathbf{Y}_j) = \frac{1}{l_{alg}} \|\mathbf{V}_x^T \mathbf{X}_i \mathbf{W}_{x_i}^{alg} - \mathbf{V}_y^T \mathbf{Y}_j \mathbf{W}_{y_j}^{alg}\|_F, \tag{3.21}$$

where the alignment step $l_{alg}$ was used to normalize the distance. Given the distance computed in Eq. 3.21, classification for a test sequence was done finding the closest to the training sequence. The overall classification error was averaged over all testing sequences.

Fig. 3.8a-b display the $12$-by-$12$ distance matrices computed by DTW and GCTW respectively. Each element in the matrices encodes the distance between a training sequence (row) and a testing sequence (column). We re-ordered the rows and columns of the matrices so that the

sequences containing the same activities were grouped in consecutive rows and columns. We then divided the matrix into nine $4$-by-$4$ blocks (yellow lines), where the block in the $i^{th}$ row and $j^{th}$ column contains the distances between the training sequences of the $i^{th}$ action and the testing data of the $j^{th}$ action, where $i, j \in \{\text{walk}, \text{run}, \text{jump}\}$. Darker color denotes smaller distance. Ideally, if the distance would be able to capture perfectly the activity, the matrix will be perfect with zeros (black color) in the diagonal blocks and higher values (white color) elsewhere.

From Fig. 3.8a, we can see that the DTW distance values in the first and last row of blocks were smaller than the blocks in the middle row. That is, DTW cannot encode well the distance of running. In comparison, Fig. 3.8b shows that GCTW captured better the distance between actions. Fig. 3.8c shows the nearest-neighbor classification error using different distances. Overall, CTW and GCTW achieved lower errors than others due to their feature selection in aligning videos with different features.

In the second example, we recognized actions from motion capture sequences $\{\mathbf{Y}_j\}_j$ given a training set containing of videos with different visual features $\{\mathbf{X}_i\}_i$. From the Weizmann dataset [80] we selected $24$ videos of three actions (walking, running and jumping). From the CMU motion capture database we selected $30$ sequences of subjects performing the same three actions (walking, running and jumping). For the mocap data $\mathbf{Y}$, we computed the quaternions for the $20$ joints that describe the body configuration, while each video frame $\mathbf{X}$ was encoded as a binary silhouette feature. The experimental setting was similar to the previous experiments. We divided all sequences into two disjoint sets used for training and testing respectively. We classified each testing motion capture sequence as the label of the video $\mathbf{X}$ in the training set with the smallest distance (Eq. 3.21). As shown in Fig. 3.8f, GCTW achieved the lowest classification error compared to other methods. This was because our GCTW distance captures between multimodal similarity between actions in videos.

### 3.5.6 Aligning facial expression sequences

In the fourth experiment, we compared CTW and GCTW in the task of aligning unscripted facial expression sequences. The facial videos were taken from the RU-FACS database [16], which contains digitized videos of $29$ young adults. They were recorded during an interview (approxi-

mately two-minutes long) in which they either lied or told the truth in response to an interviewer's questions. Pose orientation was mostly frontal with small to moderate out-of-plane head motion. The action units (AUs) in this database have been manually coded, and we randomly cropped video segments containing AU12 (smiling) for our experiments. Each event of AU12 is coded at its peak position. We used a person-specific active appearance model [131] to track $66$ landmarks on the face. For the alignment of AU12, we used only the $18$ landmarks that correspond to the outline of the mouth. See Fig. 3.9a for example frames aligned by GCTW where the mouth outlines are plotted.

The performance of CTW and GCTW were compared with DTW, DDTW and IMW. We initialized IMW, CTW and GCTW using the same uniform warping. Fig. 3.9b shows the alignment result obtained by different methods, where the three dimensions correspond to the first three principal components of the original signals. As an approximate ground-truth, the position of the peak frame of each AU12 event is indicated as the red and blue points on the curves in Fig. 3.9b and the intersection of the two dashed lines in Fig. 3.9c. As we can see from Fig. 3.9b-c, the two peaks in the low-dimensional projection found by CTW and GCTW are closer to the manually labeled peak than the ones in the original space used for DTW and DDTW. Finally, the distance between the peak point and the warping path is computed to quantitatively measure the performance. Fig. 3.9d shows the average error as the distance normalized by the sequence lengths over $20$ random repetitions, where CTW and GCTW achieved better performance.

### 3.5.7 Aligning large-scale motion capture sequences

This experiment illustrates the benefits of using GCTW for aligning two large-scale motion capture sequences. The two sequences were taken from the CMU-Multimodal Activity Dataset [54], which contains multi-sensor recordings (video, audio, motion capture data and accelerometers) of naturalistic behavior of $40$ subjects cooking five different recipes. The two sequences used in this experiment contain $44387$ and $48724$ frames respectively of two subjects cooking brownies. See Fig. 3.10c for several key-frames of these two sequences. For each motion capture frame, we computed the quaternions of the four joints on the right hand, resulting in a $12$-D feature vector that describes the body configuration. In this experiment, we only tested the performance

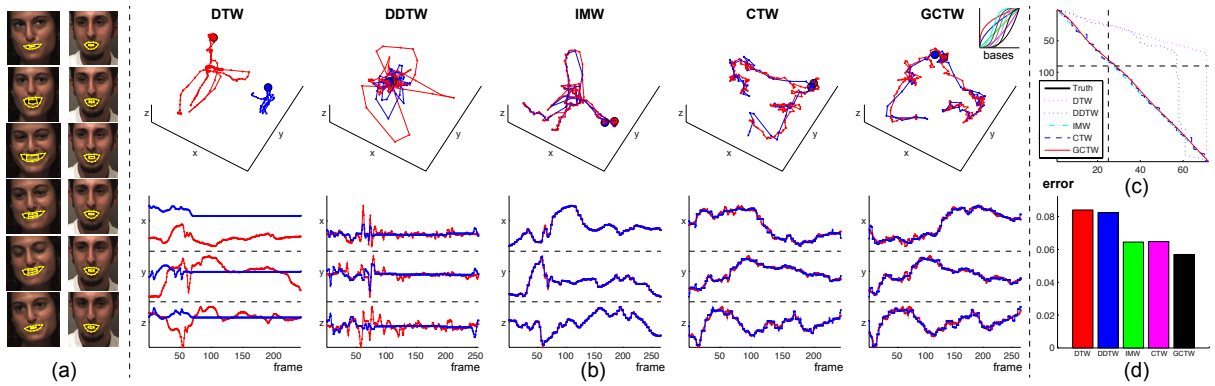


Figure 3.9: Comparison of temporal alignment algorithms for aligning facial expression sequences between different people. (a) An example of two smiling expression sequences aligned by GCTW. The features of the two sequences are computed as the 18 landmark coordinates of the mouth given by a face tracker. (b) Alignment results. The position that corresponds to the peak of the expression is indicated by points on the curves in the top row. (c) Comparison of time warping paths. The position that corresponds to the peak of the expression is indicated by the intersection of the two dashed lines. (d) Alignment errors.

of GCTW on aligning large-scale sequences, and we did not optimize GCTW over its spatial component. We used five polynomial functions and five $\tanh(\cdot)$ functions as monotonic bases for the time warping function (upper-right corner in Fig. 3.10b).

To avoid local minima in the alignment, we used a temporal coarse-to-fine strategy for the Gauss-Newton optimization in GCTW. As shown in Fig. 3.10a, the coarse-to-fine strategy proceeds in two steps: (1) In the pre-processing step, we obtained a three-level pyramid for each time series by recursively applying Gaussian smoothing with $\sigma = 200$. For instance, the first row of Fig. 3.10b illustrates the two sequences in three levels, where the ones in the first level correspond to the original signals, while the ones in the third level contain less detailed but much smoother signals. (2) In the optimization step, GCTW was first used to align the two sequences on the third level instead of the first level. The computed time warping result was then used to initialize GCTW on the second level. We repeated the same procedure to compute the final time warping result of the original sequences on the first level.

For this large-scale example, DTW was too slow and expensive to compute. However,

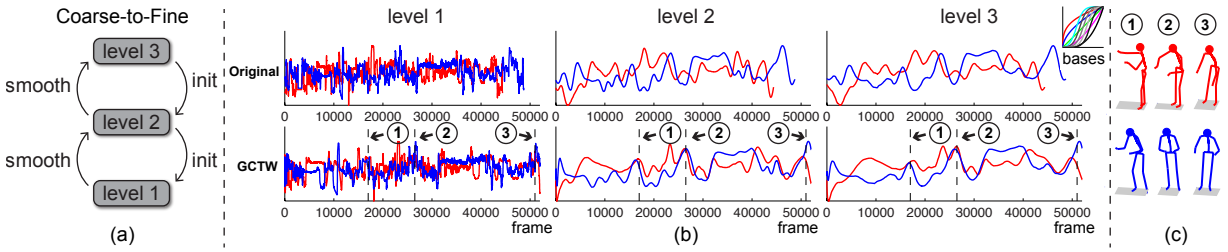


Figure 3.10: Aligning two large-scale motion capture sequences using GCTW. (a) A coarse-to-fine strategy for improving the optimization performance of GCTW. (b) The first row shows the first principal components of the original sequences for three levels of the temporal pyramids. The second row corresponds to the aligned sequences using GCTW. (c) Key frames of similar body poses aligned by GCTW.

GCTW was able to efficiently find the temporal correspondence between the sequences in just a few seconds using Matlab on a regular laptop with a 2.5GHz Intel CPU. Since the ground-truth is unknown, we qualitatively evaluated GCTW by showing the aligned key frames in Fig. 3.10c. Although the two subjects spent different amounts of time and followed different procedures in cooking, GCTW was able to align similar body poses.

### 3.5.8 Detection and alignment of similar sub-sequences

A major problem of DTW-type techniques when aligning long sequences is that they require an exact matching between the first frame and the last one, see boundary conditions (Eq. 3.5). These boundary conditions are impractical and very restrictive when only a subset of the input sequence is similar to the sequence to be aligned. Fig. 3.11 illustrates this problem: how can we align four motion capture signals composed by different walking cycles? This problem is related to sub-sequence DTW [181] and temporal commonality discovery [46]. A major limitation of these methods is its inability to handle multiple sequences. This experiment shows how can we use GCTW for multiple sub-sequence alignment in the context of aligning motion capture data.

We selected four walking sequences from the CMU motion capture database. For each motion capture frame, we computed the quaternions for 14 joints on the body, resulting in a 42-D feature vector that describes the human pose. Fig. 3.11a illustrates the first three principal com-

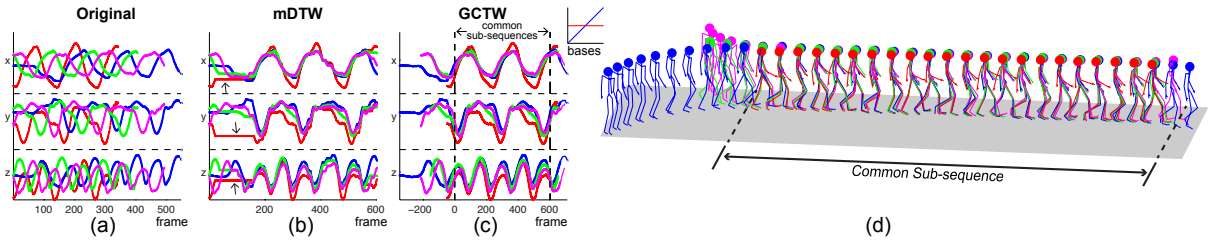


Figure 3.11: Locating and aligning similar sub-sequences of four walking motion capture signals. (a) Original features of four mocap walking sequences. (b) Alignment achieved by mDTW. mDTW tries to align the sequences end-to-end and it has to stretch some parts of the sequences (flat lines indicated by arrows). (c) Alignment by GCTW. GCTW efficiently aligns the sub-sequences and also finds the boundaries of the sub-sequences containing similar motions. (d) Key frames aligned by GCTW.

ponents of the walking sequences. To allow for sub-sequence alignment, the warping path in GCTW is represented by a combination of a constant function and a linear one as the monotonic bases (see upper-left corner of Fig. 3.11c). Both GCTW and the baseline mDTW method are initialized by uniformly aligning the sequences.

A visual comparison between mDTW and GCTW is illustrated in Fig. 3.11. Without any manual cropping, most of the conventional DTW-based methods, such as mDTW, aligned the sequences by matching the first and the last frame, which results in incorrect alignments (see Fig. 3.11b). Some parts (noted by arrows) of the sequences with fewer cycles have to be stretched into flat lines in order to match the other sequences with more cycles. Unlike conventional DTW-based methods built on dynamic programming, GCTW uses the Gauss-Newton method, which allows for a more flexible time warping. By incorporating a constant function in the set of bases, GCTW can naturally be generalized to deal with the sub-sequence alignment problem across multiple sequences. As shown in Fig. 3.11c-d, GCTW is not only able to align the sequences in time, but also locate the boundaries of the sub-sequences that contain similar motions. This experiment demonstrates the benefits of GCTW in controlling the warping path.

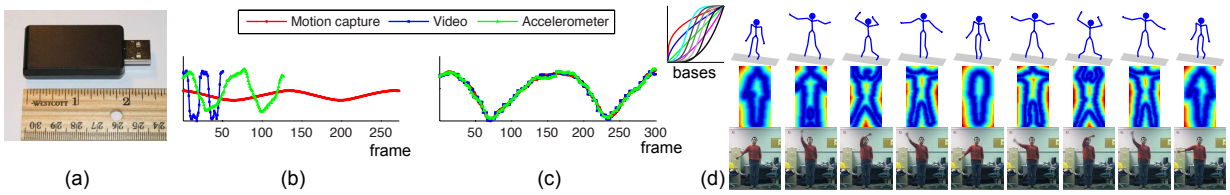


Figure 3.12: Example of aligning multi-modal sequences. (a) Accelerometer. (b) Projection onto the first principal component for the motion capture data, video and accelerometers respectively. (c) GCTW. (d) Key frames aligned by GCTW. Notice that the similar hand gestures have been aligned. From the top to bottom, we show mocap data, video, and accelerometer data respectively.

### 3.5.9 Aligning multi-modal sequences

The last experiment evaluates CTW and GCTW in the task of aligning multi-modal sequences recorded with different sensors. We selected one motion capture sequence from the CMU motion capture database, one video sequence from the Weizmann database [80], and we collected an accelerometer signal of a subject performing jumping jacks. Some instances of the multi-modal data can be seen in Fig. 3.12d. Note that, to make the problem more challenging, the two subjects in the mocap (top row) and video (middle row) sequences are performing the same activity, but in the accelerometer sequence (bottom row) the subject only moves one hand and not the legs. Even in this challenging scenario, GCTW is able to solve for the temporal correspondence that maximizes the correlation between signals. For the mocap data, we computed the quaternions for the 20 joints. In the case of the Weizmann dataset, we computed the Euclidean distance transform as described earlier. The X, Y and Z axis accelerometer data was collected using an X6-2 mini USB accelerometer (Fig. 3.12a) at a rate of 40Hz. GCTW was initialized by uniformly aligning the three sequences. We used five hyperbolic tangent and five polynomial functions as monotonic bases. Fig. 3.12b shows the first components of the three sequences projected separately by PCA. As shown in Fig. 3.12c, GCTW found an accurate temporal correspondence between the three sequences. Unfortunately, we do not have ground-truth for this experiment. However, visual inspection of the video suggests that the results are consistent with human labeling. Fig. 3.12d shows several frames that have been put in correspondence by GCTW.

## 3.6 Conclusions

This chapter proposes CTW and GCTW, two new techniques for spatio-temporal alignment of multiple multi-modal time series. CTW extends DTW by adding a feature selection mechanism and enabling alignment of signals with different dimensionality. CTW extends CCA by adding temporal alignment and allowing temporally local projections. To improve the efficiency of CTW, allow a more flexible time-warping, and align multiple sequences, GCTW extends CTW by parameterizing the warping path as a combination of monotonic functions. Inspired by existing work on image alignment, GCTW is optimized using coarse-to-fine Gauss-Newton updates, which allows for efficient alignment of long sequences.

Although CTW and GCTW have shown promising preliminary results, there are still unresolved issues. First, the Gauss-Newton algorithm used in GCTW for time warping converges poorly in areas where the objective function is non-smooth. Second, both CTW and GCTW are subject to local minima. The effect of local minima can be partially alleviated using a temporal coarse-to-fine approach as in the case of image alignment. In future work, we also plan to explore better initialization strategies. Third, although the experiments show good results using manually designed bases, we plan to learn a set of monotonic bases that are adapted to the particular alignment problem. Alternatively, a trivial extension by kernealizing the approach is expected to improve the temporal alignment error.

# Chapter 4

# Spatio-temporal Matching

## 4.1 Introduction

Human pose detection and tracking in videos have received significant attention in the last few years due to the success of Kinect cameras and applications in human computer interaction (*e.g.*, [172]), surveillance (*e.g.*, [22]) and marker-less motion capture (*e.g.*, [201]). While there have been successful methods that estimate 2D body pose from a single image [13, 63, 69, 90, 209], detecting and tracking body configurations in unconstrained video is still a challenging problem. The main challenges stem from the large variability of people's clothes, articulated motions, occlusions, outliers and changes in illumination. More importantly, existing extensions of 2D methods [69, 209] cannot cope with large pose changes due to camera view change. A common strategy to make these 2D models view-invariant is to gather and label human poses across all possible viewpoints. However, this is impractical, time consuming, and it is unclear how the space of 3D poses can be uniformly sampled. To address these issues, this chapter proposes to formulate the problem of human body detection and tracking as one of spatio-temporal matching (STM) between 3D models and video. Our method solves for the correspondence between a 3D motion capture model and trajectories in video. The main idea of our approach is illustrated in Fig. 4.1.

Our STM algorithm has two main components: (1) a spatio-temporal motion capture model that can model the configuration of several 3D joints for a variety of actions, and (2) an efficient

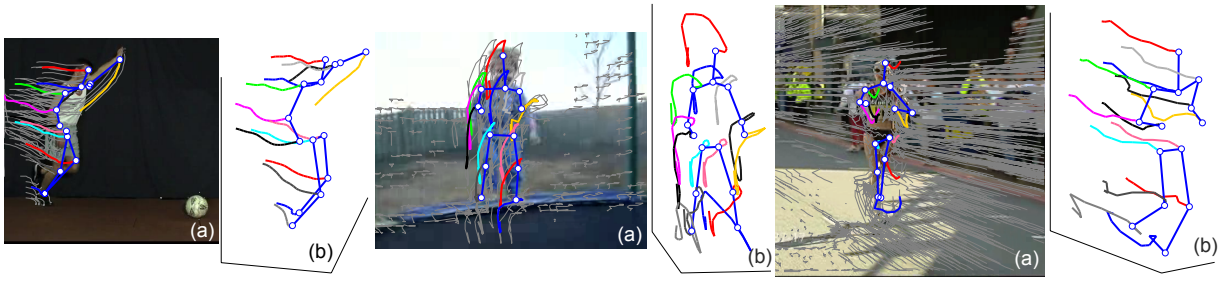


Figure 4.1: Detection and tracking of humans in three videos using spatio-temporal matching (STM). STM extracts trajectories in video (gray lines) and selects a subset of trajectories (a) that match with the 3D motion capture model (b) learned from the CMU motion capture data set. Better viewed in color.

algorithm that solves the correspondence between image trajectories and the 3D spatio-temporal motion capture model. Fig. 4.1 illustrates examples of how we can rotate our motion capture data model to match the trajectories of humans in video across several views. Moreover, our method selects a subset of trajectories that corresponds to 3D joints in the motion capture data model (about $2-4\%$ of the trajectories are selected). As we will illustrate with the Berkeley MHAD database [140], the Human3.6M database [91] and Multi-modal action dataset (MAD) [89], the main advantage of our approach is that it is able to cope with large variations in viewpoint and speed of the action. This property stems from the fact that we use 3D models.

The structure of the chapter is organized as follows. Section 4.2 reviews related work in human detection and 3D human pose estimation. Section 4.3 introduces a trajectory-based video representation. Section 4.4 discusses a spatio-temporal bilinear shape model. Section 4.5 illustrates the main idea of STM. Section 4.6 compares STM with state-of-the-art methods on several benchmark datasets. Preliminary version of this chapter was published in [225].

## 4.2 Related work

This section reviews related works in human detection in video and 3D human pose estimation.

### 4.2.1 Human detection in video

A review of the literature on people tracking is well beyond the scope of this paper. We focus our attention here on the work most similar in spirit to ours. Many early approaches [27, 56, 57, 96, 159, 173] were based on simple appearance models (*e.g.*, silhouettes) and performed tracking using stochastic search with kinematic constraints. However, silhouette extraction becomes unreliable because of complex backgrounds, occlusions, and moving cameras. Moreover, stochastic search in these high-dimensional spaces is notoriously difficult.

Facilitated by the advances in human detection methods [13, 69, 90, 162, 209], tracking by detection has been a focus of recent work. For instance, Andriluka *et al.* [11, 12] combined the initial estimate of the human pose across frames in a tracking-by-detection framework. Sapp *et al.* [163] coupled locations of body joints within and across frames from an ensemble of tractable sub-models. Wu and Nevatia [207] propose an approach for detecting and tracking partially occluded people using an assembly of body parts. Such tracking-by-detection approaches are attractive because they can avoid drift and recover from errors. The most similar work to ours are the recent fusion method by stitching together N-best hypotheses from frames of a video. Burgos *et al.* [33] merged multiple independent pose estimates across space and time using a non-maximum suppression. Park and Ramanan [143] generated multiple diverse high-scoring pose proposals from a tree-structured model and used a chain CRF to track the pose through the sequence. Compared to these methods, our work enforces temporal consistency by matching video trajectories to a spatio-temporal 3D model.

### 4.2.2 3D human pose estimation

Our method is also related to the work on 3D human pose estimation. Conventional methods rely on discriminative techniques that learn mappings from image features (*e.g.*, silhouettes [4]) to 3D pose with different priors [65, 189]. However, many of them require an accurate image segmentation to extract shape features or precise initialization to achieve good performance in the optimization. Inspired by recent advances in 2D human pose estimation, current works focus on retrieving 3D poses from 2D body part positions estimated by the off-the-shelf detectors [69,

162, 209]. For instance, Sigal and Black [174] learned a mixture of experts model to infer 3D poses conditioned on 2D poses. Simo-Serra *et al.* [175] retrieved 3D poses from the output of 2D body part detectors by a robust sampling strategy. Ionescu *et al.* [90] reconstructed 3D human pose by inferring over multiple human localization hypotheses on images. Inspired by [211], Yu *et al.* [212] recently combined human action detection and a deformable part model to estimate 3D poses. Compared to our approach, however, these methods typically require large training sets to model the large variability of appearance of different people and viewpoints.

## 4.3 Trajectory-based video representation

In order to generate candidate positions for human body parts, we used a trajectory-based representation of the input video. To be robust to large camera motion and viewpoint changes, we extracted trajectories from short video segments. The input video is temporally split into overlapped video segments of length $n$ frames (*e.g.*, $n = 15$ in all our experiments).

For each video segment, we used [198] to extract trajectories by densely sampling feature points in the first frame and track them using a dense optical flow algorithm [66]. The output of the tracker for each video segment is a set of $m_p$ trajectories,

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^1 & \cdots & \mathbf{p}_{m_p}^1 \\ \vdots & \ddots & \vdots \\ \mathbf{p}_1^n & \cdots & \mathbf{p}_{m_p}^n \end{bmatrix} \in \mathbb{R}^{2n \times m_p},$$

where each $\mathbf{p}_j^i \in \mathbb{R}^2$ denotes the 2D coordinates of the $j^{th}$ trajectory in the $i^{th}$ frame. Notice that the number of trajectories ($m_p$) can be different between segments. Fig. 4.2b illustrates a video segment with densely extracted feature trajectories.

Compared to the sparser KLT-based trackers [130, 133], densely tracking the feature points guarantees a good coverage of foreground motion and improves the quality of the trajectories in the presence of fast irregular motions. Compared to the various spatio-temporal descriptors (*e.g.*, STIP [108], Cuboids [58]), trajectories capture more local motion information of the video. see [197] for a review.

To evaluate a pseudo-likelihood of each trajectory belonging to a 3D joint, we applied a

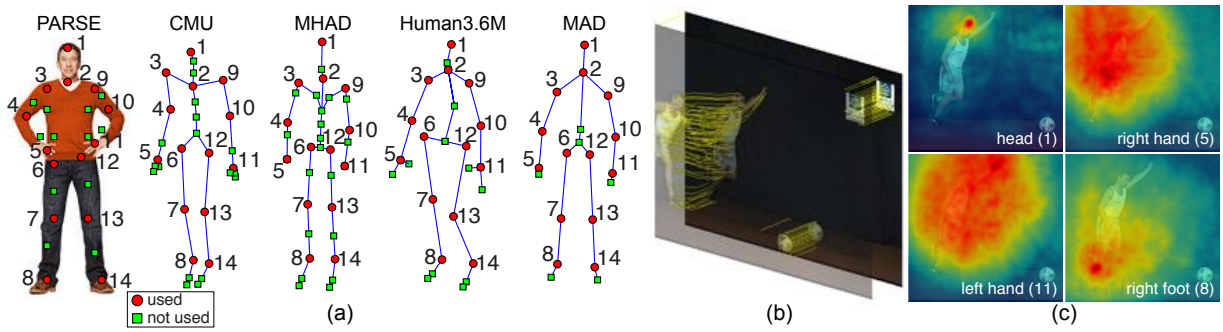


Figure 4.2: Example of feature trajectories and their responses. (a) Geometrical configuration of 14 body joints shared across 3D datasets. (b) Dense trajectories extracted from a video segment. (c) Feature response maps for 4 joints (see bottom-right corner).

state-of-the-art body part detector [209] independently on each frame. We selected a subset of $m_q = 14$ body joints (Fig. 4.2a) that are common across several datasets including the PARSE human body model [209], CMU [39], Berkeley MHAD [140], Human3.6M [91] motion capture datasets and CMU MAD Kinect dataset [89].

For each joint $c = 1 \cdots m_q$ in the $i^{th}$ frame, we computed the SVM score $a^i_{cj}$ for each trajectory $j = 1 \cdots m_p$ by performing an efficient two-pass dynamic programming inference [143]. Fig. 4.2c shows the response maps associated with four different joints. The head can be easily detected, while other joints are more ambiguous. Given a video segment containing $m_p$ trajectories, we then computed a trajectory response matrix, $\mathbf{A} \in \mathbb{R}^{m_q \times m_p}$, whose element $a_{cj} = \sum_{i=1}^{n} a^i_{cj}$ encodes the cumulative cost of assigning the $j^{th}$ trajectory to the $c^{th}$ joint over the $n$ frames.

## 4.4 Learning spatio-temporal bilinear bases

There exists a large body of work that addresses the representation of time-varying spatial data in several computer vision problems (*e.g.*, non-rigid structure from motion, face animation), see [30]. Common models include learning linear basis vectors independently for each frame [28] or discrete cosine transform bases independently for each joint trajectory [6]. Despite its simplicity, using a shape basis or a trajectory basis independently fails to exploit spatio-

temporal regularities. To have a low-dimensional model that exploits correlations in space and time, we parameterize the 3D joints in motion capture data using a bilinear spatio-temporal model [7].

Given a set of 3D motion capture sequences of different lengths, we randomly select a large number ($> 200$) of temporal segments of the same length, where each segment denoted by $\mathbf{Q}$,

$$\mathbf{Q} = \begin{bmatrix} \mathbf{q}_1^1 & \cdots & \mathbf{q}_{m_q}^1 \\ \vdots & \ddots & \vdots \\ \mathbf{q}_1^n & \cdots & \mathbf{q}_{m_q}^n \end{bmatrix} \in \mathbb{R}^{3n \times m_q},$$

contains $n$ frames and $m_q$ joints. For instance, Fig. 4.3a shows a set of motion capture segments randomly selected from several kicking sequences.

To align the segments, we apply Procrustes analysis [81] to remove the 3D rigid transformations. In order to build local models, we cluster all segments into $k$ groups using spectral clustering [136]. The affinity between each pair of segments is computed as,

$$\kappa_{ij} = \exp\Big( -\frac{1}{\sigma^2}(\|\mathbf{Q}_i - \tau_{ij}(\mathbf{Q}_j)\|_F^2 + \|\mathbf{Q}_j - \tau_{ji}(\mathbf{Q}_i)\|_F^2)\Big),$$

where $\tau_{ij}(\cdot)$ denotes the similarity transformation found by Procrustes analysis when aligning $\mathbf{Q}_j$ towards $\mathbf{Q}_i$. The kernel bandwidth $\sigma$ is set to be the average distance from the $50\%$ closest neighbors for all $\mathbf{Q}_i$ and $\mathbf{Q}_j$ pairs. As shown in the experiments, this clustering step improves the generalization of the learned shape models. For instance, each of the $4$ segment clusters shown in Fig. 4.3b corresponds to a different temporal stage of kicking a ball. Please refer Fig. 4.4 for more examples of temporal clusters.

Given a set of $l$ segments[1], $\{\mathbf{Q}_i\}_{i=1}^l$, belonging to each cluster, we learn a bilinear model [7] such that each segment $\mathbf{Q}_i$ can be reconstructed using a set of weights $\mathbf{W}_i \in \mathbb{R}^{k_t \times k_s}$ minimizing,

$$\min_{\mathbf{T},\mathbf{S},\{\mathbf{W}_i\}_i} \sum_{i=1}^{l} \|\mathcal{Q}(\mathbf{T}\mathbf{W}_i\mathbf{S}^T) - \mathbf{Q}_i\|_F^2, \tag{4.1}$$

where the columns of $\mathbf{T} \in \mathbb{R}^{n \times k_t}$ and $\mathbf{S} \in \mathbb{R}^{3m_q \times k_s}$ contain $k_t$ trajectories and $k_s$ shape bases respectively. In the experiment, we found choosing the value of $k_t$ and $k_s$ that preserve $95\%$

[1]To simplify the notation, we do not explicitly specify the cluster membership of the motion capture segment ($\mathbf{Q}_i$) and the bilinear bases ($\mathbf{T}$ and $\mathbf{S}$).

energy of the singular values, produced consistently good results. $\mathcal{Q}(\cdot)$ is a linear operator[2] that reshapes any $n$-by-$3m_q$ matrix to a $3n$-by-$m_q$ one, *i.e.*,

$$\mathcal{Q}\Big(\begin{bmatrix} \mathbf{q}_1^{1T} & \cdots & \mathbf{q}_{m_q}^{1T} \\ \vdots & \ddots & \vdots \\ \mathbf{q}_1^{nT} & \cdots & \mathbf{q}_{m_q}^{nT} \end{bmatrix}\Big) = \begin{bmatrix} \mathbf{q}_1^{1} & \cdots & \mathbf{q}_{m_q}^{1} \\ \vdots & \ddots & \vdots \\ \mathbf{q}_1^{n} & \cdots & \mathbf{q}_{m_q}^{n} \end{bmatrix}, \forall\, \mathbf{q}_j^i \in \mathbb{R}^3.$$

Unfortunately, optimizing Eq. 4.1 jointly over the bilinear bases $\mathbf{T}$, $\mathbf{S}$ and their weights $\{\mathbf{W}_i\}_i$ is a non-convex problem. To reduce the complexity and make the problem more trackable, we fix $\mathbf{T}$ to be the discrete cosine transform (DCT) bases (Top of Fig. 4.3c). Following [7], the shape bases $\mathbf{S}$ can then be computed in closed-form using the SVD as,

$$[\mathbf{T}\mathbf{T}^\dagger\mathcal{Q}^{-1}(\mathbf{Q}_1); \cdots ; \mathbf{T}\mathbf{T}^\dagger\mathcal{Q}^{-1}(\mathbf{Q}_l)] = \mathbf{U}\mathbf{\Sigma}\mathbf{S}^T. \tag{4.2}$$

For example, the left part of Fig. 4.3c plots the first two shape bases $\mathbf{s}_i$ learned from the $3^{rd}$ cluster of segments shown in Fig. 4.3b, which mainly capture the deformation of the movements of the arms and legs.

## 4.5 Spatio-temporal matching (STM)

This section describes the objective function and the optimization strategy for the STM algorithm.

### 4.5.1 STM's Objective

Given the $m_p$ trajectories $\mathbf{P} \in \mathbb{R}^{2n\times m_p}$ extracted from an $n$-length video segment, STM aims to select a subset of $m_q$ trajectories that best fits the learned spatio-temporal 3D shape structure ($\mathbf{T}$ and $\mathbf{S}$) projected in 2D. More specifically, the problem of STM consists in finding three variables: (1) a binary correspondence matrix $\mathbf{X} \in \{0,1\}^{m_p\times m_q}$ under the many-to-one constraint $\mathbf{X}^T\mathbf{1} = \mathbf{1}$; (2) the weights $\mathbf{W} \in \mathbb{R}^{k_t\times k_s}$ of the bilinear 3D model; and (3) a set of 3D-2D weak perspective

[2] $\mathcal{Q}(\mathbf{Q})$ can be written in matrix form as $\big((\mathbf{1}_{n\times m_q}\otimes\mathbf{I}_3)\circ(\mathbf{Q}\otimes\mathbf{1}_3)\big)(\mathbf{I}_{m_q}\otimes\mathbf{1}_3)$ for any $\mathbf{Q}\in\mathbb{R}^{n\times 3m_q}$.

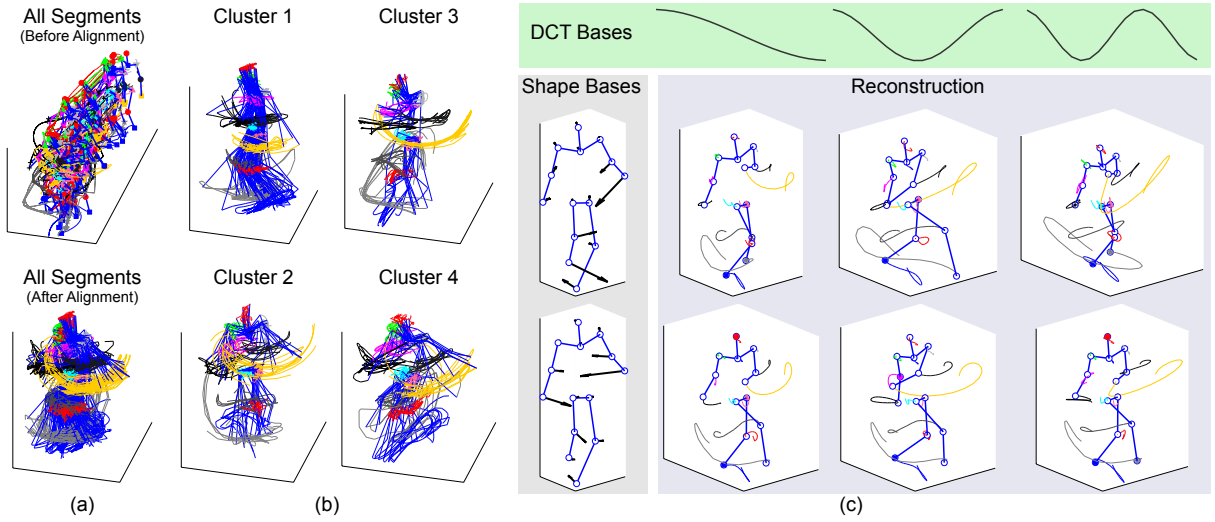


Figure 4.3: Spatio-temporal bilinear model learned from the CMU motion capture dataset. (a) Top: All the motion capture segments randomly selected from a set of kicking sequences. Bottom: The segments are spatially aligned via Procrustes alignment. (b) Clustering motion capture segments into $4$ temporal clusters. (c) The bilinear bases estimated from the $3^{rd}$ cluster. Left: top-2 shape bases ($\mathbf{s}_i$) where the shape deformation is visualized by black arrows. Top: top-3 DCT trajectory bases ($\mathbf{t}_j$). Bottom-right: bilinear reconstruction by combining each pair of shape and DCT bases ($\mathbf{t}_j\mathbf{s}_i^T$).

projections[3] $\mathbf{R} \in \mathbb{R}^{2n\times 3n}$, $\mathbf{b} \in \mathbb{R}^{2n}$, where the rotation needs to satisfy the orthogonal constraints,

$$\Psi = \left\{ \mathbf{R} = \begin{bmatrix} \theta_1\mathbf{R}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \theta_n\mathbf{R}_n \end{bmatrix} \;\middle|\; \mathbf{R}_i^T\mathbf{R}_i = \mathbf{I}_2 \;\forall\, i \right\}. \tag{4.3}$$

In a nutshell, STM aims to solve the following problem

$$\begin{aligned} \min_{\mathbf{X},\mathbf{W},\mathbf{R},\mathbf{b}} \quad & \|\mathbf{R}\mathcal{Q}(\mathbf{T}\mathbf{W}\mathbf{S}^T) + \mathbf{b}\mathbf{1}^T - \mathbf{P}\mathbf{X}\|_1 + \lambda_a \operatorname{tr}(\mathbf{A}\mathbf{X}) \\ & + \lambda_s\|\mathbf{T}\mathbf{W}\mathbf{\Sigma}^{-1}\|_1 + \lambda_o\|\mathbf{G}\mathbf{P}\mathbf{X} - \mathbf{G}'\mathbf{P}'\mathbf{X}'\|_1, \\ \text{s.t. } \mathbf{X} \in & \{0,1\}^{m_p\times m_q},\ \mathbf{X}^T\mathbf{1} = \mathbf{1}, \mathbf{R} \in \Psi, \end{aligned} \tag{4.4}$$

[3] $\mathbf{R} = [\searrow_i \theta_i\mathbf{R}_i] \in \mathbb{R}^{2n\times 3n}$ is a block-diagonal matrix, where each block contains the rotation $\mathbf{R}_i \in \mathbb{R}^{2\times 3}$ and scaling $\theta_i$ for each frame. Similarly, $\mathbf{b} = [\mathbf{b}_1; \cdots; \mathbf{b}_n] \in \mathbb{R}^{2n}$ is a concatenation of the translation $\mathbf{b}_i \in \mathbb{R}^2$ for each frame.

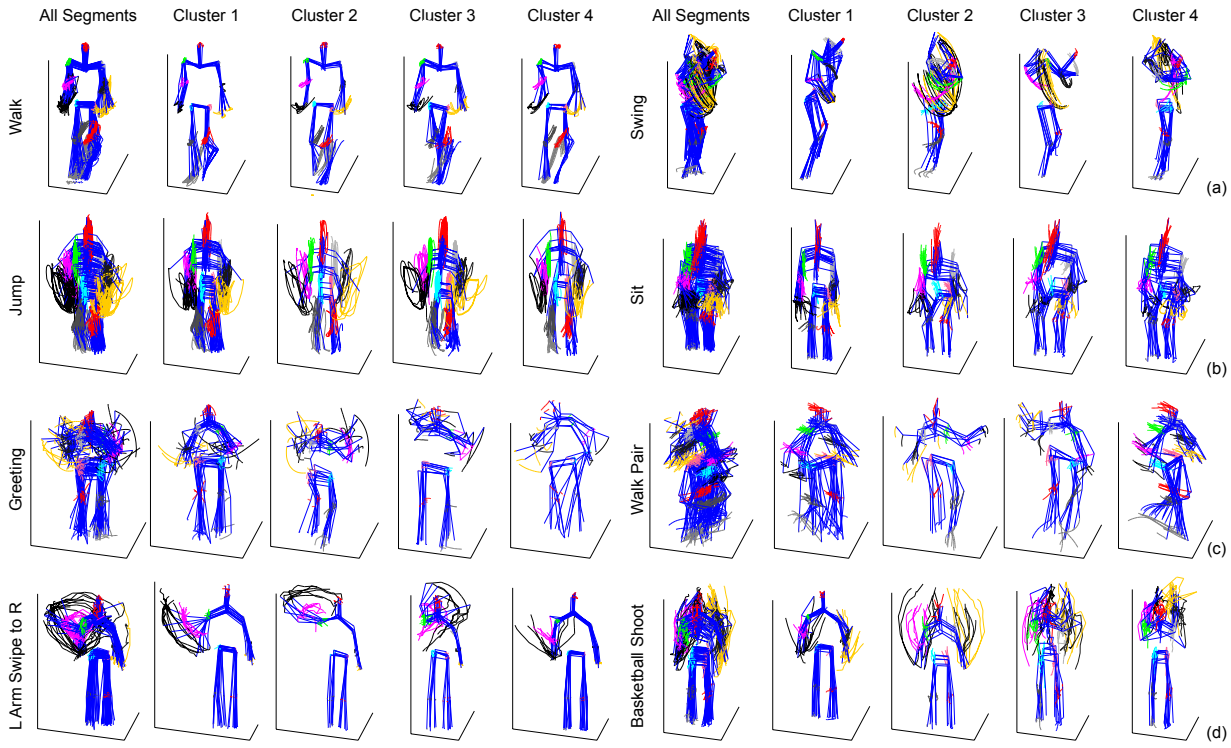


Figure 4.4: Clustering motion capture segments into four clusters for different datasets. (a) CMU motion capture dataset [39]. (b) Berkeley MHAD dataset [140]. (c) Human3.6M dataset [91]. (d) CMU MAD dataset [89].

where the objective is composed by four terms. (1) The first term measures the error between the selected trajectories $\mathbf{PX} \in \mathbb{R}^{2n \times m_q}$ and the bilinear reconstruction $\mathcal{Q}(\mathbf{TWS}^T)$ projected in 2D using $\mathbf{R}$ and $\mathbf{b}$. The error is computed using the $l_1$ norm instead of the Frobenious norm, because of its efficiency and robustness. (2) Given the trajectory response $\mathbf{A} \in \mathbb{R}^{m_q \times m_p}$, the second term measures the appearance cost of the trajectories selected by $\mathbf{X}$ and weighted by $\lambda_a$. (3) The third term weighted by $\lambda_s$ penalizes large weights $\mathbf{TW} \in \mathbb{R}^{n \times k_s}$ of the shape bases, where the singular value $\mathbf{\Sigma} \in \mathbb{R}^{k_s \times k_s}$ computed in Eq. 4.2 is used to normalize the contribution of each basis. (4) To impose temporal continuity on the solution, the fourth term weighted by $\lambda_o$ penalizes the $l_1$ distance between the reconstruction for the current segment $\mathbf{PX}$ and the previous one $\mathbf{P'X'}$, where $\mathbf{G} \in \{0,1\}^{2n_o \times 2n}$ and $\mathbf{G'} \in \{0,1\}^{2n_o \times 2n'}$ are two selection matrices that select the overlapped $n_o$ frames between $\mathbf{PX}$ and $\mathbf{P'X'}$ respectively. In our experiment, the regularization weights $\lambda_a$, $\lambda_s$ and $\lambda_o$ are estimated using cross-validation.

Optimizing Eq. 4.4 is a challenging problem[4], in the following sections we describe an efficient coordinate-descent algorithm that alternates between solving $\mathbf{X}, \mathbf{W}$ and $\mathbf{R}, \mathbf{b}$ until convergence. The algorithm is initialized by computing $\mathbf{X}$ that minimizes the appearance cost $\mathrm{tr}(\mathbf{AX})$ in Eq. 4.4 and setting $\mathcal{Q}(\mathbf{TWS}^T)$ to be the mean of the motion capture segments.

### 4.5.2 Optimizing STM over $\mathbf{X}$ and $\mathbf{W}$

Due to the combinatorial constraint on $\mathbf{X}$, optimizing Eq. 4.4 over $\mathbf{X}$ and $\mathbf{W}$ given $\mathbf{R}$ and $\mathbf{b}$ is a NP-hard mixed-integer problem. To approximate the problem, we relax the binary $\mathbf{X}$ to be a continuous one and reformulate the problem using the LP trick[5] [94] as,

$$\begin{aligned}
\min_{\substack{\mathbf{X},\mathbf{W},\mathbf{U},\mathbf{V}\\ \mathbf{U}_s,\mathbf{V}_s,\mathbf{U}_o,\mathbf{V}_o}} \quad & \mathbf{1}^T(\mathbf{U}+\mathbf{V})\mathbf{1} + \lambda_a \,\mathrm{tr}(\mathbf{AX}) + \lambda_s \mathbf{1}^T(\mathbf{U}_s+\mathbf{V}_s)\mathbf{1} + \lambda_o \mathbf{1}^T(\mathbf{U}_o+\mathbf{V}_o)\mathbf{1}, \qquad (4.5)\\
\text{s.t.} \quad & \mathbf{X} \in [0,1]^{m_p \times m_q}, \mathbf{X}^T\mathbf{1} = \mathbf{1},\\
& \mathbf{R}\mathcal{Q}(\mathbf{TWS}^T) + \mathbf{b}\mathbf{1}^T - \mathbf{PX} = \mathbf{U} - \mathbf{V},\ \mathbf{U},\mathbf{V} \geq \mathbf{0},\\
& \mathbf{TW}\boldsymbol{\Sigma}^{-1} = \mathbf{U}_s - \mathbf{V}_s,\ \mathbf{U}_s,\mathbf{V}_s \geq \mathbf{0},\\
& \mathbf{GPX} - \mathbf{G}'\mathbf{P}'\mathbf{X}' = \mathbf{U}_o - \mathbf{V}_o,\ \mathbf{U}_o,\mathbf{V}_o \geq \mathbf{0},
\end{aligned}$$

where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{2n \times m_q}$ and $\mathbf{U}_s, \mathbf{V}_s \in \mathbb{R}^{n \times k_s}$ are four auxiliary variables used to formulate the $l_1$ problem as linear programming. The term $\mathbf{R}\mathcal{Q}(\mathbf{TWS}^T)$ is linear in $\mathbf{W}$ and we can conveniently re-write this expression using the following equality as:

$$\begin{aligned}
&\mathrm{vec}\left(\mathbf{R}\mathcal{Q}(\mathbf{TWS}^T)\right) = (\mathbf{I}_{m_q} \otimes \mathbf{R})\,\mathrm{vec}\left(\mathcal{Q}(\mathbf{TWS}^T)\right)\\
&= (\mathbf{I}_{m_q} \otimes \mathbf{R})\boldsymbol{\Pi}_{\mathcal{Q}}\,\mathrm{vec}(\mathbf{TWS}^T) = \underbrace{(\mathbf{I}_{m_q} \otimes \mathbf{R})\boldsymbol{\Pi}_{\mathcal{Q}}(\mathbf{S} \otimes \mathbf{T})}_{\text{Constant}}\,\mathrm{vec}(\mathbf{W}),
\end{aligned}$$

[4]Although there are some works that can remove the explicit optimization over rotation by computing the Laplacian matrix [129], it is difficult to use it in our work because the Laplacian matrix is not invariant to $3D$-$2D$ projection.

[5]Given an LP problem with absolute value in the objective, *e.g.*, $\min_x \|x\|$, we can equivalently solve, $\min_{x,u,v} u + v$, by introducing two positive auxiliary variables $u, v \geq 0$ with the constraint $x = u - v$.

where $\mathbf{\Pi}_{\mathcal{Q}} \in \{0,1\}^{3nm_q \times 3nm_q}$ is a permutation matrix that re-orders the elements of a $3nm_q$-D vector as,

$$\mathbf{\Pi}_{\mathcal{Q}} \operatorname{vec}\Big( \begin{bmatrix} \mathbf{q}_1^1 & \cdots & \mathbf{q}_{m_q}^1 \\ \vdots & \ddots & \vdots \\ \mathbf{q}_1^n & \cdots & \mathbf{q}_{m_q}^n \end{bmatrix} \Big) = \operatorname{vec}\Big( \begin{bmatrix} \mathbf{q}_1^{1T} & \cdots & \mathbf{q}_{m_q}^{1T} \\ \vdots & \ddots & \vdots \\ \mathbf{q}_1^{nT} & \cdots & \mathbf{q}_{m_q}^{nT} \end{bmatrix} \Big).$$

After solving the linear program, we gradually discretize $\mathbf{X}$ by taking successive refinements based on trust-region shrinking [94].

### 4.5.3 Optimizing STM over $\mathrm{R}$ and $\mathrm{b}$

If $\mathbf{X}$ and $\mathbf{W}$ are fixed, optimizing Eq. 4.4 with respect to $\mathbf{R}$ and $\mathbf{b}$ becomes an $l_1$ Procrustes problem [186],

$$\min_{\mathbf{R},\mathbf{b}} \quad \|\mathbf{RQ} + \mathbf{b1}^T - \mathbf{PX}\|_1, \quad \text{s.t. } \mathbf{R} \in \Psi, \tag{4.6}$$

where $\mathbf{Q} = \mathcal{Q}(\mathbf{TWS}^T)$. Inspired by the recent advances in compressed sensing, we approximate Eq. 4.6 using the augmented Lagrange multipliers method [118] that minimizes the following augmented Lagrange function:

$$\min_{\mathbf{L},\mathbf{E},\mu,\mathbf{R},\mathbf{b}} \|\mathbf{E} - \mathbf{PX}\|_1 + \operatorname{tr}\Big(\mathbf{L}^T(\mathbf{RQ} + \mathbf{b1}^T - \mathbf{E})\Big) + \frac{\mu}{2}\|\mathbf{RQ} + \mathbf{b1}^T - \mathbf{E}\|_F^2, \quad \text{s.t. } \mathbf{R} \in \Psi, \tag{4.7}$$

where $\mathbf{L}$ is the Lagrange multiplier, $\mathbf{E}$ is an auxiliary variable, and $\mu$ is the penalty parameter. Eq. 4.7 can be efficiently approximated in a coordinate-descent manner. First, optimizing Eq. 4.7 with respect to $\mathbf{R}$ and $\mathbf{b}$ is a standard orthogonal Procrustes problem,

$$\min_{\mathbf{R},\mathbf{b}} \quad \|\mathbf{RQ} + \mathbf{b1}^T - (\mathbf{E} - \frac{\mathbf{L}}{\mu})\|_F^2, \quad \text{s.t. } \mathbf{R} \in \Psi,$$

which has a close-form solution using the SVD. Second, optimizing Eq. 4.7 with respect to $\mathbf{E}$ can be efficiently found using absolute value shrinkage [118] as,

$$\mathbf{E} = \mathbf{PX} - \mathcal{S}_{\frac{1}{\mu}}(\mathbf{PX} - \mathbf{RQ} - \mathbf{b1}^T - \frac{\mathbf{L}}{\mu}),$$

where $\mathcal{S}_\sigma(p) = \max(|p| - \sigma, 0)\,\mathrm{sign}(p)$ is a soft-thresholding operator [118]. Third, we gradually update $\mathbf{L} \leftarrow \mathbf{L} + \mu(\mathbf{RQ} + \mathbf{b1}^T - \mathbf{E})$ and $\mu \leftarrow \rho\mu$, where we set the incremental ratio to $\rho = 1.05$ in all our experiments.

### 4.5.4 Fusion

Given a video containing an arbitrary number of frames, we solved STM independently for each segment of $n$ frames ($n = 15$ in our experiments). Recall that we learned $k$ bilinear models ($\mathbf{T}$ and $\mathbf{S}$) from different clusters of motion capture segments (*e.g.*, Fig. 4.3b) in the training step. To find the best model for each segment, we optimize Eq. 4.4 using each model and select the one with the objective.

After solving STM for each segment, we need to aggregate the local solutions to generate the global one. Specifically, how to generate the coordinate $\bar{\mathbf{P}}^i \in \mathbb{R}^{2\times m_q}$ at $i^{th}$ frame from the selected trajectories $\{\mathbf{P}_c\mathbf{X}_c\}_c$ of $l_c$ segments overlapped at $i^{th}$ frame. In the following, we explore two ways to compute $\bar{\mathbf{P}}^i$.

**Averaging.** The first solution is to average the coordinates of the selected trajectories $\{\mathbf{P}_c\mathbf{X}_c\}_c$ overlapped at $i$,

$$\bar{\mathbf{P}}^i = \frac{1}{l_c}\sum_c \mathbf{P}_c^{i_c}\mathbf{X}_c, \tag{4.8}$$

where $\mathbf{P}_c^{i_c} \in \mathbb{R}^{2\times m_p}$ encodes the trajectory coordinates at the $i_c^{th}$ frame within the $c^{th}$ segment and $i_c$ the local index of the $i^{th}$ frame in the original video.

**Winner-Take-All.** In the second way, we evaluate the objective value (Eq. 4.4) of each local solution $\{\mathbf{P}_c\mathbf{X}_c\}_c$ marginalized at the overlapped frame $i$, *i.e.*,

$$J_c = \|\left[\mathbf{R}\mathcal{Q}(\mathbf{TWS}^T) + \mathbf{b1}^T - \mathbf{PX}\right]_{i_c}\|_1 + \lambda_a\|\left[\mathbf{AX}\right]_{i_c}\|_1 + \lambda_s\|\left[\mathbf{TW\Sigma}^{-1}\right]_{i_c}\|_1, \tag{4.9}$$

where the operator $[\cdot]_{i_c}$ is to take the rows of a matrix associated to $i_c^{th}$ frame. Once the $J_c$s for each segment are computed, we pick the one $c^* = \arg\min_c J_c$ with the minimum objective value as the final solution $\bar{\mathbf{P}}^i = \mathbf{P}_{c^*}^{i_c^*}\mathbf{X}_{c^*}$.

## 4.6 Experiments

This section compares STM against several state-of-the-art algorithms for body part detection in synthetic experiments on the CMU motion capture dataset [39], and real experiments on the MHAD [140], the Human3.6M [91] and the MAD [89] datasets.

For each dataset, the 3D motion capture model was trained from its associated motion capture sequences. The 3D motion capture training data is person-independent, and it does not contains samples of the testing subject. Notice that the annotation scheme is different across datasets (Fig. 4.2a). We investigated four different types of 3D models for STM: (1) Generic models: **STM-G1** and **STM-G4** were trained using all sequences of different actions with $k = 1$ and $k = 4$ clusters respectively. (2) Action-specific models: **STM-A1** and **STM-A4** were trained independently for each action from each dataset. In testing, we assumed we know what action the subject was performing. As before, **STM-A1** and **STM-A4** were trained with $k = 1$ and $k = 4$ clusters respectively. In addition to the different choices of 3D models, we also testified the effect of using different fusion methods. The postfix notations, **-AVE** and **-WTA** stand for using the averaging and winner-take-all methods in the fusion step respectively.

### 4.6.1 CMU motion capture dataset

The first experiment validated our approach on the CMU motion capture dataset [39], from which we selected $5$ actions including walking, running, jumping, kicking, golf swing. For each action, we picked $8$ sequences performed by different subjects. For each sequence, we synthetically generated $0 \sim 200$ random trajectories as outliers in 3D. Then we projected each sequence (with outliers included) onto $4$ different 2D views. See Fig. 4.5a for examples of the 3D sequences as well as the camera positions. To reproduce the response of a body part detector at each frame, we synthetically generate a constant-value response region centered at the ground-truth location with the radius being the maximum limb length over the sequence. The response value of the $j^{th}$ feature trajectory for the $c^{th}$ body part at $i^{th}$ frame is considered to be $a^i_{cj} = -1$ if it falls in the region or $0$ otherwise. Our goal is to detect the original trajectories and recover the body structure.

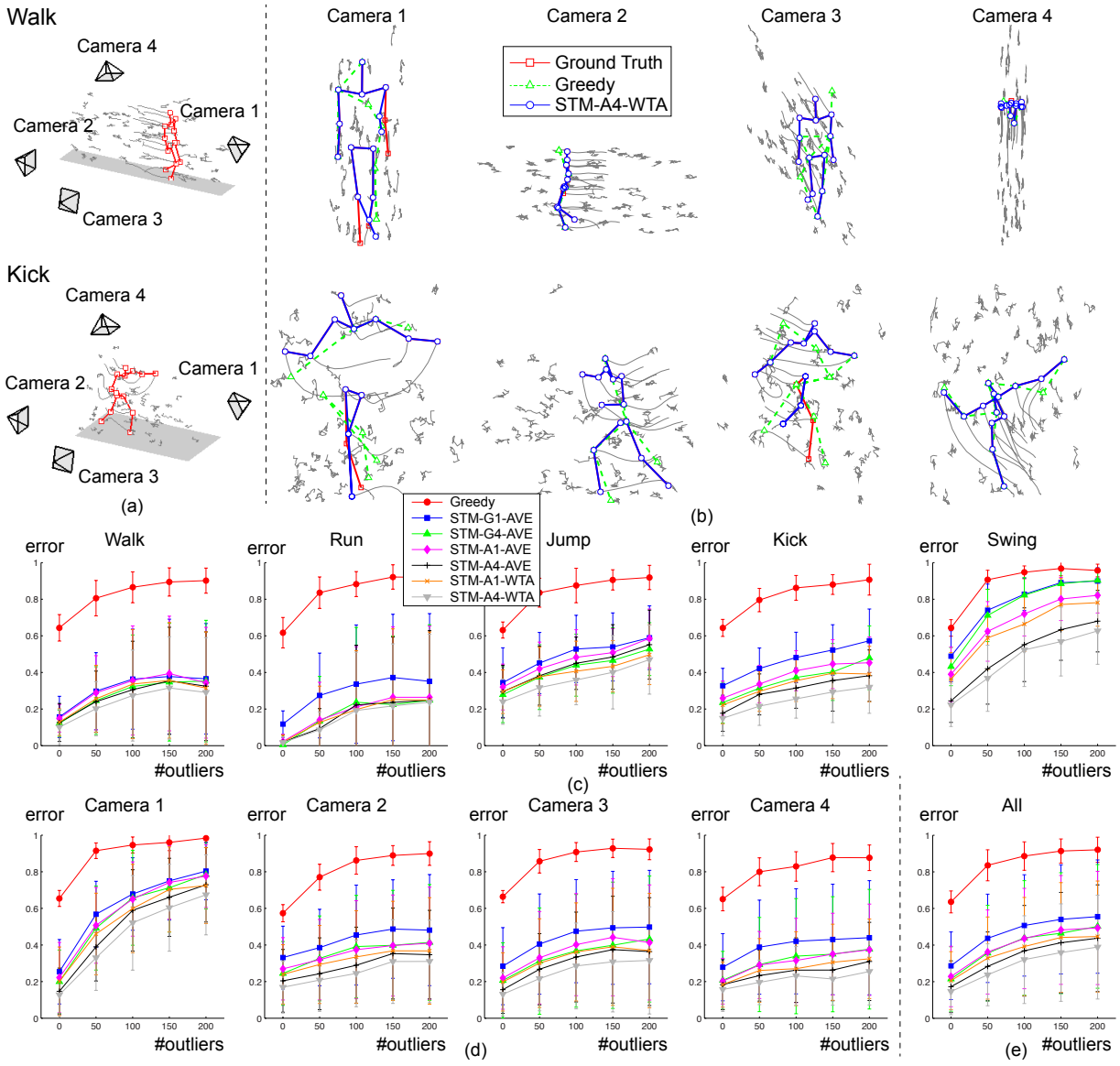


Figure 4.5: Comparison of human pose estimation on the CMU motion capture dataset. (a) Original motion capture key-frames in 3D with 50 outliers that were synthetically generated. (b) Results of the greedy approach and our method on four 2D projections. (c) Mean error and std. for each method and action as a function of the number of outliers. (d) Mean error and std. for each camera view. (e) Mean error and std. for all actions and cameras.

We quantitatively evaluated our method with a leave-one-out scheme, *i.e.*, each testing sequence was taken out for testing, and the remaining data was used for training the bilinear model. For each sequence, we computed the error of each method as the percentage of incorrect detections of the feature points compared with the ground-truth position averaged over frames. To the best of our knowledge, there is no previous work on STM in computer vision. Therefore, we implemented a greedy baseline that selects the optimal feature points with the lowest response cost without geometrical constraints.

Fig. 4.5b shows some key-frames for the greedy approach, our method and the ground truth using the STM-A4-WTA for detecting the kicking actions across four views. As can be observed, STM is able to select the trajectories more precisely and it is more robust than the greedy approach. Fig. 4.5c-d quantitatively compare our methods with the greedy approach on each action and viewpoint respectively. Our method consistently outperforms the greedy approach for detection and tracking in presence of outliers. In addition, the STM-A1-AVE model obtains lower error rates than STM-G1-AVE because STM-A1-AVE is an action-specific model, unlike STM-G1-AVE which is a generic one. By increasing the number of clusters from one to four, the performance of STM-G4-AVE and STM-A4-AVE clearly improves from STM-G1-AVE and STM-A1-AVE respectively. This not surprising because the bilinear models trained on a group of similar segments can be represented more compactly (fewer number of parameters) and generalize better in testing. In addition, using the winner-take-all method (STM-A1-WTA and STM-A4-WTA) in the fusion step can further improve the performance compared to the averaging approaches (STM-A1-AVE and STM-A4-AVE). This is because the winner-take-all method picks the best local solution at the overlapped region, yielding more robust solution in the case when a large number of outliers exist.

### 4.6.2 Berkeley multi-modal human action dataset (MHAD)

In the second experiment, we tested the ability of STM to detect humans on the Berkeley multi-modal human action database (MHAD) [140]. The MHAD database contains 11 actions performed by 12 subjects. For each sequence, we took the videos captured by 2 different cameras as shown in Fig. 4.6a. To extract the trajectories from each video, we used [198] in sliding-window

manner to extract dense trajectories from each 15 frames segment. The response for each trajectory was computed using the SVM detector score [209]. The bilinear models were trained from the motion capture data associated with this dataset.

To quantitatively evaluate the performance, we compared our method with two baselines: the state-of-the-art image-based pose estimation method proposed by Yang and Ramanan [209], and the two recent video-based methods designed by Park and Ramanan [143] and Burgos *et al.* [33] that merge multiple independent pose estimates across frames. We evaluated all methods with a leave-one-out scheme. The error for each method is computed as the pixel distance between the estimated and ground-truth part locations. Notice that a portion of the error is due to the inconsistency in labeling protocol between the PARSE model [209] and the MHAD dataset.

Fig. 4.6b-d compare the error to localize body parts of our method against [33, 143, 209]. Our method largely improves the image-based baseline [209] for all actions and viewpoints. Compared to the video-based method [33], STM achieves lower errors for most actions except for "jump jacking", "bending", "one-hand waving" and "two-hand waving", where the fast movement of the body joints cause much larger error in tracking feature trajectories over time. Among the four STM models, STM-A4-AVE performs the best because the clustering step improves the generalization of the bilinear model. In this experiment, however, we found taking the average coordinates of the local solutions in fusion steps (STM-A1-AVE and STM-A4-AVE) yielded lower error than the ones (STM-A1-WTA and STM-A4-WTA) using the winner-take-all mechanism. This is because the noise (*e.g.*, drifting and missing point) generated in the dense tracking step can be mitigated by the averaging step. As shown in Fig. 4.6d, the hands are the most difficult to accurately detect because of their fast movements and frequent occlusions.

Fig. 4.6e-g investigate the three main parameters of our system, segment length ($n$), number of bases ($k_s$ and $k_t$) and the regularization weights ($\lambda_a$ and $\lambda_s$). According to Fig. 4.6e, a segment length is beneficial for "jump jacking" because the performance of the tracker [198] is less stable for fast-speed action. In contrast, using a larger window improves the temporal consistency in actions such as "throwing" and "standing up". Fig. 4.6f shows the detection error of STM using different number of shape ($k_s$) and trajectories ($k_t$) bases for the first subject. Overall, we found the performance of STM is not very sensitive to small change in the number of shape

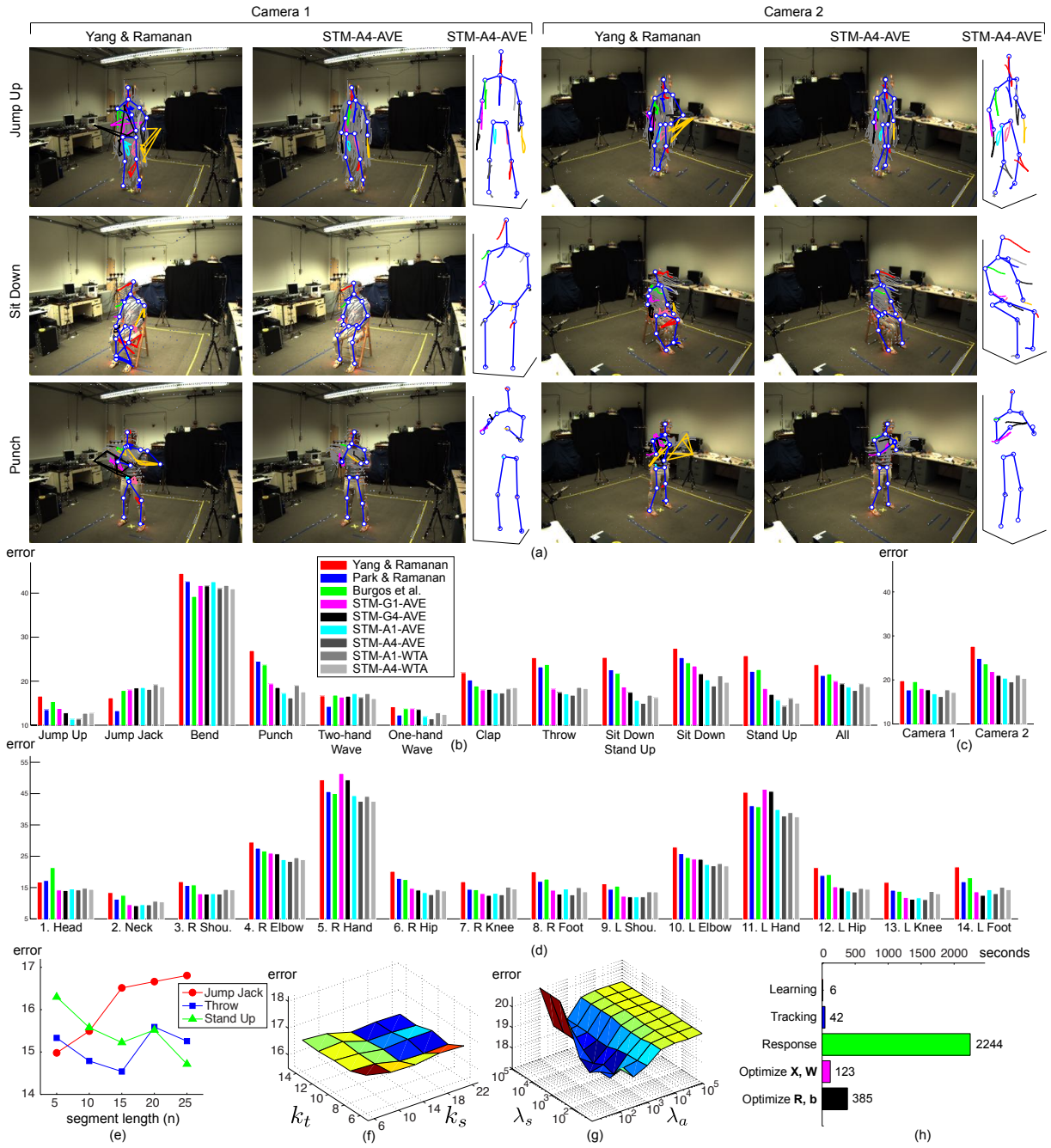


Figure 4.6: Comparison of human pose estimation on the Berkeley MHAD dataset. (a) Result of [209] and our method on three actions of two views, where the 3D reconstruction estimated by our method is plotted on the right. (b) Errors for each action. (c) Errors for each camera view. (d) Errors of each joint. (e) Errors with respect to the segment length ($n$). (f) Errors with respect to the bases number ($k_s$ and $k_t$). (g) Errors with respect to the regularization weights ($\lambda_a$ and $\lambda_s$). (h) Time cost of each step.

bases because the contribution of each shape basis in STM (Eq. 4.4) is by their energies ($\mathbf{\Sigma}$). In addition, using a number (*e.g.*, 5) of trajectory bases can lower the performance of STM. This result demonstrates the effectiveness of using dynamic models over the static ones (*e.g.*, a PCA-based model can be considered as a special case of the bilinear model when $k_t = 1$). Fig. 4.6g plots the cross-validation error for the first subject, from which we pick the optimal $\lambda_a$ and $\lambda_s$.

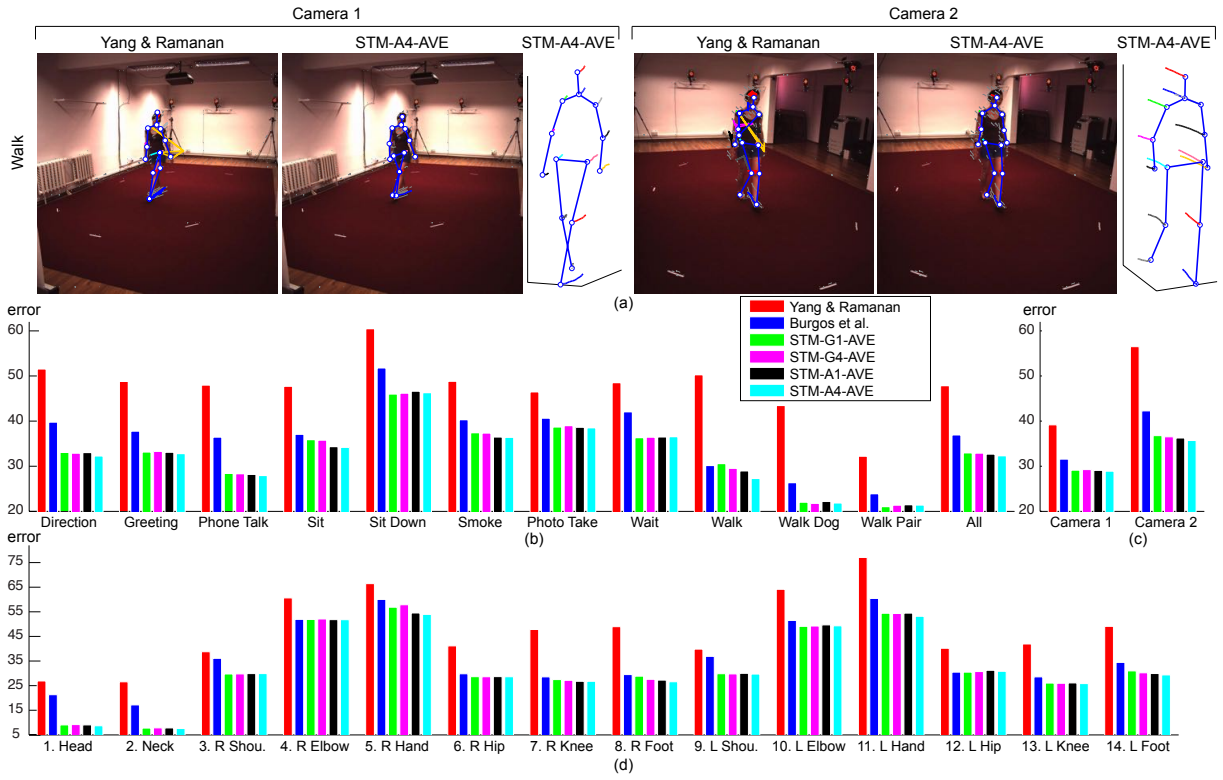


Figure 4.7: Comparison of human pose estimation on the Human 3.6M dataset. (a) Result of [209] and our method on three actions of two views, where the 3D reconstruction estimated by our method is plotted on the right. (b) Errors for each action. (c) Errors for each camera view. (d) Errors of each joint.

Our system was implemented in Matlab on a PC with 2GHz Intel CPU and 8GB memory. The codes of [33, 209] were downloaded from authors' webpages. The linear programming in Eq. 4.5 was optimized using the Mosek LP solver [134]. Fig. 4.6f analyzes the computational cost (in seconds) for tracking the human pose in a sequence containing 126 frames. The most computationally intensive part of the method is calculating the response for each joint and each

frame using [209]. Despite a large number of candidate trajectories ($m_p \approx 700$) per segment, STM can be computing in about $8$ minutes.

### 4.6.3 Human3.6M dataset

In the last experiment, we selected $11$ actions performed by $5$ subjects from the Human3.6M dataset [91]. Compared to the Berkeley MHAD dataset, the motions in Human3.6M were performed by professional actors, that wear regular clothing to maintain as much realism as possible. See Fig. 4.7a for example frames.

As in the previous experiment, our methods were compared with two baselines [33, 209] in a leave-one-out scheme. The bilinear models were trained from the motion capture data associated with this dataset. Fig. 4.6b-c show the performance of each method on localizing body part for each action and viewpoint respectively. Due to the larger appearance variation and more complex motion performance, the overall error of each method is larger than the one achieved on the previous Berkeley MHAD dataset. However, STM still outperforms both the baselines [33, 209] for most actions and viewpoints. If the action label is known a priori, training action-specific models (STM-A1-AVE and STM-A4-AVE) achieves better performance than the ones trained on all actions (STM-G1-AVE and STM-G4-AVE).

### 4.6.4 CMU multi-modal action dataset (MAD)

In the last experiment, we test our method on the CMU multi-modal action detection (MAD) dataset [89]. Unlike MHAD and H3M datasets where each sequence contains only one action, MAD contains $40$ sequences of $20$ subjects ($2$ sequences per subject) performing $35$ activities in each of the sequences. Therefore, this experiment can more closely reveal the performance of different human pose estimation methods in real applications. The length of each sequence is around $2-4$ minutes ($4000-7000$ frames). The 2D and 3D coordinates of $14$ joints for each frame was recorded using the Microsoft Kinect sensor in an indoor environment. The $35$ actions include full-body motion (*e.g.*, run, crouch, jump), upper-body motion (*e.g.*, throw, basketball dribble, baseball swing), and lower-body motion (*e.g.*, kick). Each subject performs all the 35

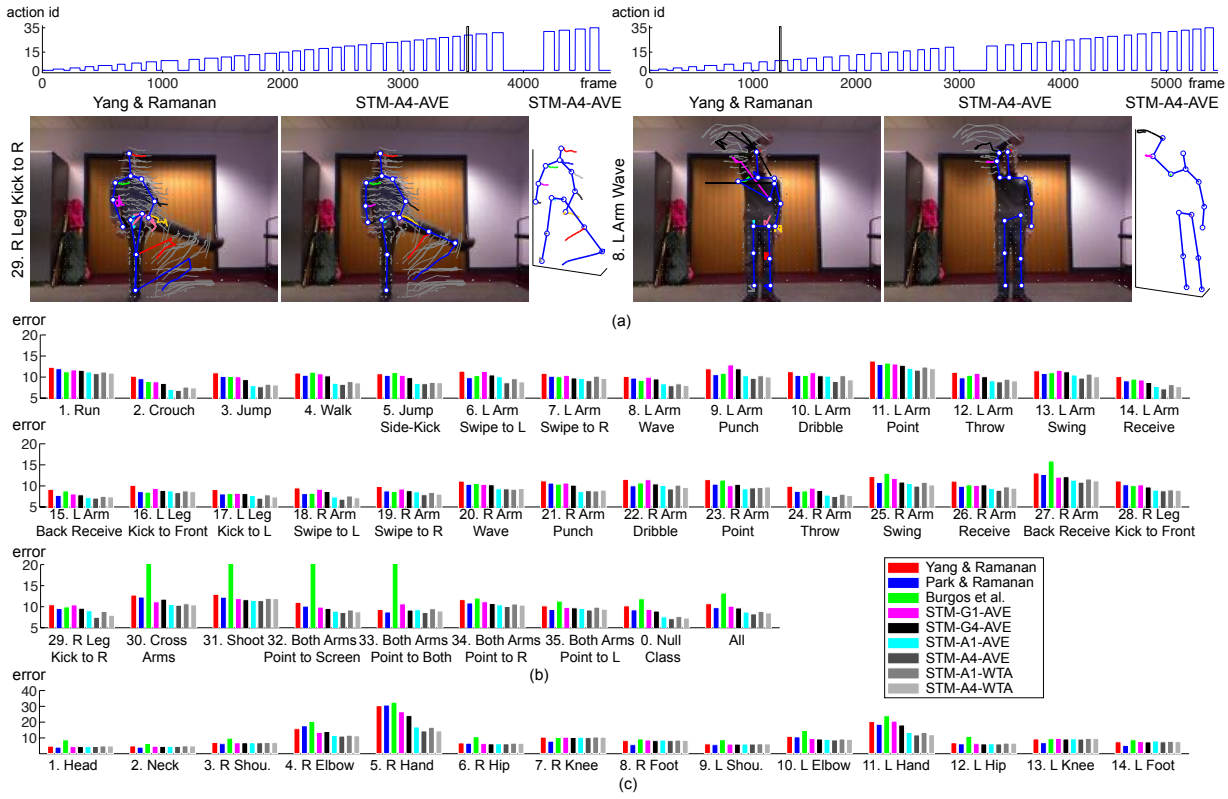


Figure 4.8: Comparison of human pose estimation on the CMU MAD dataset. (a) Top: Action id of two sequences. Bottom: Result of [209] and our method on two actions, where the 3D reconstruction estimated by our method is plotted on the right. (b) Errors for each action. (c) Errors of each joint.

activities continuously, and the segments between two actions are considered the null class (*i.e.*, the subject is standing). Fig. 4.8a show some example frames and frame labels for two sequences.

As in the previous experiment, our methods were compared with three baselines [33, 143, 209] in a leave-one-out scheme. The bilinear models were trained from the Kinect data associated with this dataset. Fig. 4.8b shows the performance of each method on localizing body part for each action. Fig. 4.8c shows the error of each joint. Compared to the baselines [33, 143, 209], STM achieved lower error in most joints, especially the hand and elbow which had larger movements than others. If the action label is known a priori, training action-specific models (STM-A1-AVE and STM-A4-AVE) achieves better performance than the ones trained on all actions (STM-G1-AVE and STM-G4-AVE). Similar to previous results, the averaging fusion

step (STM-A1-AVE and STM-A4-AVE) performed better than the winner-take-all (STM-A1-WTA and STM-A4-WTA).

## 4.7 Conclusion

This chapter presents spatio-temporal matching (STM), a robust method for detection and tracking human poses in videos by matching video trajectories to a 3D motion capture model. STM matches trajectories to a 3D model, and hence it provides intrinsic view-invariance. The main novelty of the work resides in computing the correspondence between video and motion capture data. Although it might seem computationally expensive and difficult to optimize at first, using an $l_1$-formulation to solve for correspondence results in an algorithm that is efficient and robust to outliers, missing data and mismatches. We showed how STM outperforms state-of-the-art approaches to object detection based on deformable parts models in the Berkeley MHAD [140], the Human3.6M [91] and the CMU MAD [89] datasets.

A major limitation of our current approach is the high computational cost for calculating the joint response, which is computed independently for each frame. In future work, we plan to incorporate richer temporal features [198] to improve the speed and accuracy of the trajectory response. Also, we are solving STM independently for each segment (sub-sequence), which might result in some discontinuity in the estimation of the pose; a straight-forward improvement could be made by imposing consistency between overlapping segments.

# Chapter 5

# Conclusion

In this dissertation, we focuses on solving correspondence problems in computer vision. In particular, we have developed three techniques that address the problems of spatial matching, temporal matching and spatio-temporal matching respectively. In the following, we conclude with a summary of our contributions and limitations.

## 5.1 Spatial matching

Finding spatial correspondence between image features is an essential task in recognizing actions from images and videos. Graph matching is one of the most robust methods for solving feature matching problem in computer vision. However, the computational complexity of existing methods limits the permissible size of input graphs in practice. In Chapter 2, we propose factorized graph matching (FGM), a new framework for better optimizing and understanding graph matching (GM) problems. The key idea of FGM is a novel factorization of the pairwise affinity matrix into six smaller components. Four main benefits follow from factorizing the affinity matrix. First, there is no need to explicitly compute the costly affinity matrix. Second, it allows for a unification of GM methods as well as provides a clean connection with existing iterative closest point algorithms. Third, the factorization enables the use of path-following algorithms that improve the performance of GM methods. Finally, it becomes easier to incorporate global geometrical transformation in GM.

We have illustrated the advantages of factorizing the pair-wise affinity matrix of typical graph matching problems. The most computationally consuming part of the algorithm is the large number of iterations needed for FW method to converge when $J_\alpha$ is close to a convex function. Therefore, more advanced techniques (*e.g.*, conjugate gradient) can be used to speedup FW. In addition, we are currently exploring the extension of this factorization methods to other higher-order graph matching problems [41, 60, 216] as well as learning parameters for graph matching [37, 116].

## 5.2 Temporal matching

Temporal alignment of time series is a fundamental step in many applications such as activity recognition or synthesis human motions. Major challenges to align human motion include: (1) introducing a feature selection to compensate for subjects' physical characteristics, different motion styles and speed of actions. (2) allowing alignment between different features (*e.g.*, video and mocap). Chapter 3 proposes canonical time warping (CTW) and generalized canonical time warping (GCTW), two new techniques for aligning multiple multi-modal time series. CTW extends dynamic time warping (DTW) techniques by adding a feature selection mechanism and enabling alignment of signals with different dimensionality. CTW extends canonical correlation analysis (CCA) by adding temporal alignment and allowing temporally local projections. To improve the efficiency of CTW, allow a more flexible time-warping, and align multiple sequences, GCTW extends CTW by parameterizing the warping path as a combination of monotonic functions. Inspired by existing work on image alignment, GCTW is optimized using coarse-to-fine Gauss-Newton updates, which allows for efficient alignment of long sequences.

Although CTW and GCTW have shown promising preliminary results, there are still unresolved issues. First, the Gauss-Newton algorithm used in GCTW for time warping converges poorly in areas where the objective function is non-smooth. Second, both CTW and GCTW are subject to local minima. The effect of local minima can be partially alleviated using a temporal coarse-to-fine approach as in the case of image alignment. In future work, we also plan to explore better initialization strategies. Third, although the experiments show good results using manu-

ally designed bases, we plan to learn a set of monotonic bases that are adapted to the particular alignment problem.

## 5.3 Spatio-temporal matching

Finding correspondence between spatio-temporal features tracked over video is an essential task in recognizing temporal events from videos. Chapter 4 presents spatio-temporal matching (STM), a robust method for detection and tracking human poses in videos by matching video trajectories to a 3D motion capture model. STM matches trajectories to a 3D model, and hence it provides intrinsic view-invariance. The main novelty of the work resides in computing the correspondence between video and motion capture data. Although it might seem computationally expensive and difficult to optimize at first, using an $l_1$-formulation to solve for correspondence results in an algorithm that is efficient and robust to outliers, missing data and mismatches. We showed how STM outperforms state-of-the-art approaches to object detection based on deformable parts models in the Berkeley MHAD [140], the Human3.6M [91] and the CMU MAD [89] datasets.

A major limitation of our current approach is the high computational cost for calculating the joint response, which is computed independently for each frame. In future work, we plan to incorporate richer temporal features [198] to improve the speed and accuracy of the trajectory response. In addition, we are interested in extending this work to more robustly deal with outliers and occlusions in more challenging cases.

# Chapter 6

# Future work

We have proposed in this dissertation a new graph matching method for finding the correspondence between spatial features, a new temporal matching approach for aligning multi-modal sequences with different features, and a spatio-temporal framework for matching video trajectories and 3D models. While these methods have produced promising results, solving matching problems in computer vision is still challenging. In this chapter, we discusses some potential ideas for future work.

## 6.1 Dense correspondence

In Chapter 2, we have presented an efficient graph matching method for finding correspondence between two sets of feature points. To match $n$ points, graph matching method typically needs to build a huge affinity matrix, $\mathbf{K} \in \mathbb{R}^{n^2 \times n^2}$, to encode node and edge similarities. Although our factorization idea can avoid the most expensive construction of $\mathbf{K}$, it still needs to compute a pair of node and edge affinity matrices, $\mathbf{K}_p \in \mathbb{R}^{n \times n}$ and $\mathbf{K}_q \in \mathbb{R}^{m \times m}$, where $m$ is the number of edges. Due to its quadratic complexity, graph matching is still too cumbersome to solve the dense correspondence problems. For instance, the optical flow [14] and stereo matching problems [164, 179] consists of assigning optimal discrete state to each pixel of an image with pairwise constraints (*e.g.*, local smoothness). Given a $400$-by-$300$ image of normal size, the number of nodes could reach $120,000$, which is far beyond the capacity of most graph matching

method.

To address the dense correspondence problems, we usually resort to more efficient algorithms with linear complexity in the number of nodes such as, belief propagation [203], graph cuts [24], dynamic programming [68], linear programming [195], and randomized approaches [15]. However, traditional dense correspondence methods only consider relatively simple geometric deformations (*e.g.*, 1D disparity for stereo matching and 2D translations for optical flow). Based on SIFT features, SIFTFlow [120] can robustly aligns complex scene pairs containing larger spatial differences. More recently, DeepFlow [202] handles the large translations in optical flow by using the deep network. When complex deformations exist, however, the above methods might still fail.

Allowing for more complex and non-rigid deformation, graph matching can benefit a variety of applications relying on dense correspondence methods. To scale up the graph matching method to handle tens of thousands of nodes, however, we need to improve the robustness and computational complexity of the current methods. Recently, there has been some efforts towards this goal. For instance, Cho and Lee [42] proposed a progressive framework to update candidate matches based on pair-wise geometric relationships between new matches and the current graph matching result. However, it tends to introduce many outliers because the current graph matching result might be noisy. Furthermore, its computational complexity is high because exploring the full matching space is required. One of my future goal is to integrate outlier detection and node clustering into graph matching with less computational cost.

## 6.2 Learning for matching

In this dissertation, we have presented several efficient and accurate approximations to approach the NP-hard graph matching, temporal alignment and the spatial-temporal matching problems. Although we can develop accurate approximation to these problems, a natural question still needs to be addressed for many tasks: how can we learn better parameters and models to improve the matching performance? For instance, how can we set the regularization weights in the objectives of graph matching (Eq. 2.7) and spatio-temporal matching (Eq. 4.4)? How can we obtain a better

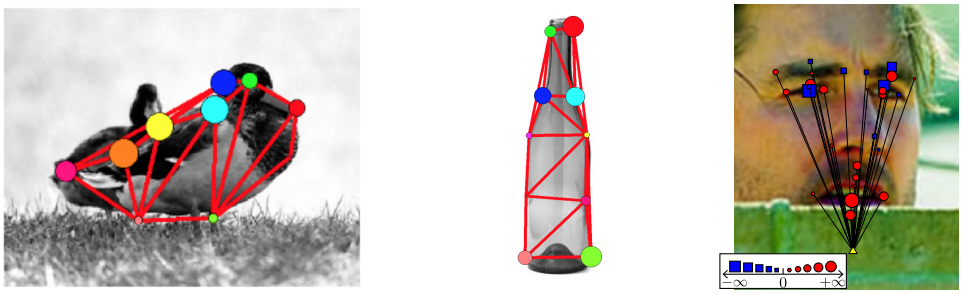


Figure 6.1: Examples of learning graph models for matching. (a-b) Learned graph model for two generic objects [44]. (c) Learned graph models for a specific object (faces) [226].

graph model for a target object (*e.g.*, the bird in Fig. 6.1a) to match?

To tackle this issue, two groups of methods have been proposed. In the first group, people have addressed the problem of learning a set of parameters that characterize similarity functions using a training set, such that the estimated optimal match on a test pair of graphs agrees with the best match. For instance, it has been shown that a better objective function for graph matching can be learned in either supervised [37] or unsupervised [116] manner. The other group of research aimed at learning a graph model instead of a better matching function. As shown in Fig. 6.1a-b, Cho *et al.* [44] proposed an effective scheme to parameterize a graph model, and learned its structure and parameters for visual object category matching. In a recent work [226], we have shown how we can learn a better graph model for matching a more specific object: faces. In this work, the local graph structure of each facial landmark is represented by a set of weights of other landmarks (Fig. 6.1c).

In the future, we will attempt to learn more sparse representations of graphs to balance between the matching performance and the complexity of graph. We hope this will lead us to more efficient construction of graph structure for a target class.

## 6.3 Fine-grained recognition

Fine-grained recognition concerns recognizing subordinate object classes, such as distinguishing different breeds of dogs [100, 122], species of birds [194], models of cars [106]. These tasks

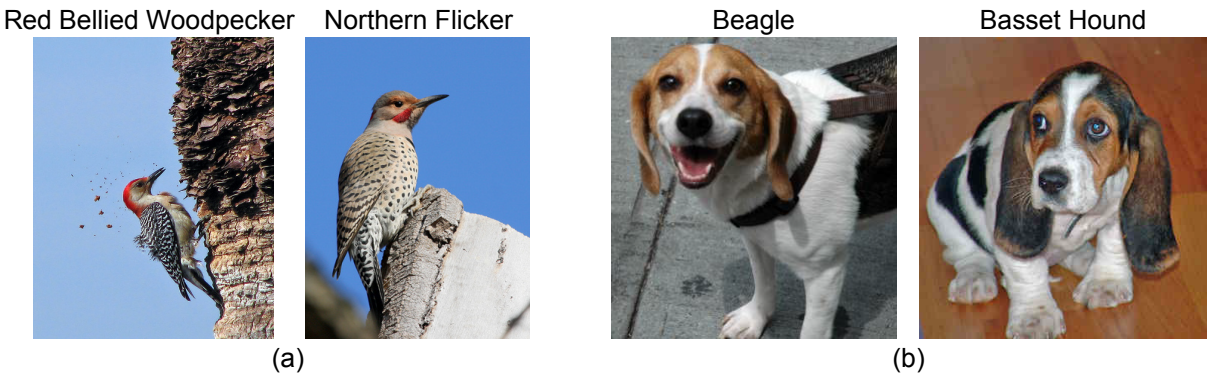


Figure 6.2: Examples of similar species for fine-grained recognition. (a) Two species of birds [194]. (b) Two species of dogs [122].

yield a great deal of information for a human user and can thus add tremendous value to society. Fine-grained recognition is challenging because fine grained labels are much harder to acquire. In addition, there are much fewer discriminative features compared to categorization at the basic level. Distinguishing a bird from a hand-grove is easy because there are plenty of helpful visual cues. As shown in Fig. 6.2, however, the difference between two species of woodpeckers (*e.g.*, Northern Flicker and Red Bel-lied Woodpecker) or dogs (*e.g.*, Beagle and Basset Hound) can be very subtle and only a few key features matter.

To tackle the challenge of feature selection, current methods can be divided into two groups: (1) One approach is applying specialized domain knowledge. For instance, Kumar *et al.* [107] built LeafSnap, a visual recognition system for automatic plant species identification. This approach yield great success, but demands from the researcher a deep understanding of the specific domain. (2) Another direction is including the crowd in the loop by having humans either label or propose parts and attributes. For instance, Farrell *et al.* [67] used poselets [23] as a part localization scheme for fine-grained categorization tasks. Zhang *et al.* [217, 218] proposed deformable part descriptors, using DPM part boxes as the building block for pose-normalized representations for fine-grained categorization task. More recently, Deng *et al.* [55] introduced Bubbles, a online game reveals discriminative features humans use. These approaches can potentially reduce the burden of domain specific engineering.

Following the previous work, one of my future direction is to introduce graph matching for

improving the performance in fine-grained object recognition. To distinguish the subtle differences between similar objects, we need to design more expressive spatial models than the traditional bags of features [177] and their variants [111]. Tree-based spatial models such as pictorial structure [68, 69, 209] have been proved to be very effective in different domains including human pose estimation. However, the tree-like structures are sometimes insufficient to express the subtle variance of object shapes and poses. In the future, I plan to explore an efficient and effective graph matching method to capture the detailed structural difference and help improving the performance and accuracy in fine-grained object recognition.

# Bibliography

[1] CMU/VASC Image Database. http://vasc.ri.cmu.edu//idb/html/motion/house/index.html.

[2] Car and motorbike image database.

[3] J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[4] A. Agarwal and B. Triggs. Recovering 3d human pose from monocular images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[5] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Computer Survey*, 43(3):16, 2011.

[6] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade. Trajectory space: A dual representation for nonrigid structure from motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(7):1442–1456, 2011.

[7] I. Akhter, T. Simon, S. Khan, I. Matthews, and Y. Sheikh. Bilinear spatiotemporal basis models. *ACM Trans. Graphics*, 31(2):17, 2012.

[8] E. L. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. SIAM, 2003.

[9] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(5): 522–525, 1993.

[10] T. W. Anderson. *An introduction to multivariate statistical analysis*. Wiley-Interscience, 2003.

[11] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[12] M. Andriluka, S. Roth, and B. Schiele. Monocular 3d pose estimation and tracking by detection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[13] M. Andriluka, S. Roth, and B. Schiele. Discriminative appearance models for pictorial structures. *Int'l J. Computer Vision*, 99(3):259–280, 2012.

[14] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *Int'l J. Computer Vision*, 92(1):1–31, 2011.

[15] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graphics*, 28(3), 2009.

[16] M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. Fasel, and J. R. Movellan. Automatic recognition of facial actions in spontaneous expressions. *J. Multimedia*, 1(6): 22–35, 2006.

[17] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[18] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[19] D. P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific, 1995.

[20] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[21] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[22] O. Boiman and M. Irani. Detecting irregularities in images and in video. *Int'l J. Computer Vision*, 74(1):17–31, 2007.

[23] L. D. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose

annotations. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

[24] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[25] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[26] M. Brand and A. Hertzmann. Style machines. In *ACM SIGGRAPH*, pages 183–192, 2000.

[27] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1998.

[28] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[29] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[30] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical geometry of non-rigid shapes*. Springer, 2008.

[31] A. Bruderlin and L. Williams. Motion signal processing. In *ACM SIGGRAPH*, pages 97–104, 1995.

[32] H. Bunke. Graph matching: Theoretical foundations, algorithms, and applications. In *Proc. Int'l Conf. Vision Interface*, 2000.

[33] X. Burgos, D. Hall, P. Perona, and P. Dollár. Merging pose estimates across space and time. In *Proc. British Machine Vision Conference*, 2013.

[34] R. Burkard, M. DellAmico, and S. Martello. *Assignment Problems*. SIAM, 2009.

[35] T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(4):515–519, 2004.

[36] T. S. Caetano, T. Caelli, D. Schuurmans, and D. Augusto. Graphical models and point pattern matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1646–1663, 2006.

[37] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(6):1048–1058, 2009.

[38] B. Caputo and L. Jie. A performance evaluation of exact and approximate match kernels for object recognition. *Electronic Letters on Computer Vision and Image Analysis*, 8(3): 15–26, 2009.

[39] Carnegie Mellon University Motion Capture Database. http://mocap.cs.cmu.edu.

[40] Y. Caspi and M. Irani. Aligning non-overlapping sequences. *Int'l J. Computer Vision*, 48 (1):39–51, 2002.

[41] M. Chertok and Y. Keller. Efficient high order matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(12):2205–2215, 2010.

[42] M. Cho and K. M. Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[43] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *Proc. European Conf. Computer Vision*, 2010.

[44] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[45] Selina Chu, Eamonn Keogh, David Hart, and Michael Pazzani. Iterative deepening dynamic time warping for time series. In *Proc. SIAM Int'l Conf. Data Mining*, 2002.

[46] W.-S. Chu, F. Zhou, and F. De la Torre. Unsupervised temporal commonality discovery. In *Proc. European Conf. Computer Vision*, 2012.

[47] H. Chui and A. Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[48] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *Int'l J. Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[49] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26

(10):1367–1372, 2004.

[50] T. Cour and J. Shi. Solving Markov random fields with spectral relaxation. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, 2007.

[51] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Proc. Neural Information Processing Systems*, 2006.

[52] A. D. J. Cross and E. R. Hancock. Graph matching with a dual-step EM algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11):1236–1253, 1998.

[53] F. De la Torre. A least-squares framework for component analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(6):1041–1055, 2012.

[54] F. De la Torre, J. K. Hodgins, J. Montano, and S. Valcarcel. Detailed human data acquisition of kitchen activities: the CMU-multimodal activity database (CMU-MMAC). Technical Report RI-TR-08-22, CMU, 2009.

[55] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[56] J. Deutscher and I. D. Reid. Articulated body motion capture by stochastic search. *Int'l J. Computer Vision*, 61(2):185–205, 2005.

[57] J. Deutscher, A. Blake, and I. D. Reid. Articulated body motion capture by annealed particle filtering. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000.

[58] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *ICCV VS-PETS*, 2005.

[59] I. L. Dryden and K. V. Mardia. *Statistical shape analysis*. Wiley, 1998.

[60] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(12): 2383–2395, 2011.

[61] O. Duchenne, A. Joulin, and J. Ponce. A graph-matching kernel for object categorization. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[62] A. Egozi, Y. Keller, and H. Guterman. A probabilistic approach to spectral graph match-

ing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(1):18–27, 2013.

[63] M. Eichner, M. Jesús, A. Zisserman, and V. Ferrari. 2d articulated human pose estimation and retrieval in (almost) unconstrained still images. *Int'l J. Computer Vision*, 99(2):190–214, 2012.

[64] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(10):1285–1295, 2003.

[65] A. M. Elgammal and C.-S. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004.

[66] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *SCIA*, 2003.

[67] R. Farrell, O. Oza, N. Zhang, V. I. Morariu, T. Darrell, and L. S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[68] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *Int'l J. Computer Vision*, 61(1):55–79, 2005.

[69] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[70] B. Fischer, V. Roth, and J. Buhmann. Time-series alignment by non-negative multiple generalized canonical correlation analysis. *BMC Bioinformatics*, 8(10), 2007.

[71] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395, 1981.

[72] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image Vision Computing*, 21(13-14):1145–1153, 2003.

[73] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[74] M. Fukushima. A modified Frank-Wolfe algorithm for solving the traffic assignment problem. *Transp. Res. Part B: Methodological*, 18(2):169–177, 1984.

[75] U. Gaur, Y. Zhu, B. Song, and A. Roy-Chowdhury. A "string of feature graphs model" for recognition of complex activities in natural videos. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[76] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[77] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(4):377–388, 1996.

[78] D. Gong and G. G. Medioni. Dynamic manifold warping for view invariant action recognition. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[79] L. Gorelick, M. Galun, E. Sharon, R. Basri, and A. Brandt. Shape representation and classification using the Poisson equation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(12):1991–2005, 2006.

[80] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[81] J. C. Gower and G. B. Dijksterhuis. *Procrustes problems*. Oxford University Press, 2004.

[82] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[83] J. Ham, D. Lee, and L. Saul. Semisupervised alignment of manifolds. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, 2005.

[84] M. A. Hasan. On multi-set canonical correlation analysis. In *Proc. Int'l Joint Conf. Neural Networks*, 2009.

[85] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *Proc. European Conf. Computer Vision*, 2006.

[86] A. Heloir, N. Courty, S. Gibet, and F. Multon. Temporal alignment of communicative

gesture sequences. *J. Visualization and Computer Animation*, 17(3-4):347–357, 2006.

[87] R. Horst and P. M. Pardalos. *Handbook of Global Optimization*. Kluwer, 1995.

[88] E. Hsu, K. Pulli, and J. Popovic. Style translation for human motion. *ACM Trans. Graphics*, 24(3), 2005.

[89] D. Huang, Y. Wang, S. Yao, and F. De la Torre. Sequential max-margin event detectors. In *Proc. European Conf. Computer Vision*, 2014.

[90] C. Ionescu, F. Li, and C. Sminchisescu. Latent structured models for human pose estimation. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[91] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014.

[92] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proc. Int'l Conf. Machine Learning*, 2013.

[93] H. Jiang and D. R. Martin. Finding actions using shape flows. In *Proc. European Conf. Computer Vision*, 2008.

[94] H. Jiang, M. S. Drew, and Z.-N. Li. Matching by linear programming and successive convexification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):959–975, 2007.

[95] Hao Jiang, Stella X. Yu, and David R. Martin. Linear scale and rotation invariant matching. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(7):1339–1355, 2011.

[96] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition*, 1996.

[97] I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-independent action recognition from temporal self-similarities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(1): 172–185, 2011.

[98] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim,

B. X. Kausler, J Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problems. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[99] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *Proc. SIAM Int'l Conf. Data Mining*, 2001.

[100] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei. Novel dataset for fine-grained image categorization. In *CVPR Workshop on Fine-Grained Visual Categorization*, 2011.

[101] T. K. Kim and R. Cipolla. Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31:1415–1428, 2009.

[102] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1568–1583, 2006.

[103] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[104] N. Komodakis, N. Paragios, and G. Tziritas. MRF energy minimization and beyond via dual decomposition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(3):531–552, 2011.

[105] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

[106] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop on 3D Representation and Recognition*, 2013.

[107] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. John, I. C. Lopez, and J. V. B. Soares. LeafSnap: A computer vision system for automatic plant species identification. In *Proc. European Conf. Computer Vision*, 2012.

[108] I. Laptev. On space-time interest points. *Int'l J. Computer Vision*, 64(2-3):107–123, 2005.

[109] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[110] E. L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

[111] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[112] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Proc. IEEE Int'l Conf. Computer Vision*, 2005.

[113] M. Leordeanu and M. Hebert. Efficient MAP approximation for dense energy functions. In *Proc. Int'l Conf. Machine Learning*, 2006.

[114] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[115] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and MAP inference. In *Proc. Neural Information Processing Systems*, 2009.

[116] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *Int'l J. Computer Vision*, 95(1):1–18, 2011.

[117] H. Li, E. Kim, X. Huang, and L. He. Object matching with a locally affine-invariant constraint. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[118] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[119] J. Listgarten, R. M. Neal, S. T. Roweis, and A. Emili. Multiple alignment of continuous time series. In *Proc. Neural Information Processing Systems*, 2005.

[120] C. Liu, J. Yuen, and A. Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[121] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos in the wild. In

*Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009.

[122] J. Liu, A. Kanazawa, D. W. Jacobs, and P. N. Belhumeur. Dog breed classification using part localization. In *Proc. European Conf. Computer Vision*, 2012.

[123] Z. Liu, H. Qiao, and L. Xu. An extended path following algorithm for graph-matching problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(7):1451–1456, 2012.

[124] E. M. Loiola, N. M. De Abreu, P. O. Boaventura, P. Hahn, and T. M. Querido. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.*, 176(2):657–690, 2007.

[125] C. C. Loy, T. Xiang, and S. Gong. Time-delayed correlation analysis for multi-camera activity understanding. *Int'l J. Computer Vision*, 90(1):106–129, 2010.

[126] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Int'l Joint Conf. Artificial Intelligence*, 1981.

[127] B. Luo and E. R. Hancock. A unified framework for alignment and correspondence. *Computer Vision and Image Understanding*, 92(1):26–55, 2003.

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

[129] D. Mateus, R. Horaud, D. Knossow, F. Cuzzolin, and E. Boyer. Articulated shape matching using Laplacian eigenfunctions and unsupervised point registration. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[130] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCVW*, 2009.

[131] I. Matthews and S. Baker. Active appearance models revisited. *Int'l J. Computer Vision*, 60(2):135–164, 2004.

[132] C. R. Maurer, R. Qi, and V. V. Raghavan. A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[133] R. Messing, C. J. Pal, and H. A. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

[134] Mosek. http://www.mosek.com/.

[135] A. Myronenko and X. B. Song. Point set registration: Coherent point drift. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.

[136] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Proc. Neural Information Processing Systems*, pages 849–856, 2001.

[137] K. M. Ng. *A continuation approach for solving nonlinear optimization problems with discrete variables*. PhD thesis, Stanford University, 2002.

[138] M. A. Nicolaou, V. Pavlovic, and M. Pantic. Dynamic probabilistic CCA for analysis of affective behavior and fusion of continuous annotations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 36(7):1299–1311, 2014.

[139] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *Int'l J. Computer Vision*, 79(3):299–318, 2008.

[140] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy. Berkeley MHAD: A comprehensive multimodal human action database. In *IEEE Workshop on Applications on Computer Vision (WACV)*, pages 53–60, 2013.

[141] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the web. Technical report, Stanford, 1998.

[142] W. Pan and L. Torresani. Unsupervised hierarchical modeling of locomotion styles. In *Proc. Int'l Conf. Machine Learning*, 2009.

[143] D. Park and D. Ramanan. N-best maximal decoders for part models. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[144] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11(8):1933–1955, 1999.

[145] ILOG CPLEX: High performance software for mathematical programming and optimization. http://www.ilog.com/products/cplex/.

[146] H. Pirsiavash and D. Ramanan. Steerable part models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[147] N. Quadrianto, A. J. Smola, L. Song, and T. Tuytelaars. Kernelized sorting. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(10):1809–1821, 2010.

[148] L. Rabiner and B. Juang. *Fundamentals of speech recognition*. Prentice Hall, 1993.

[149] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. Brandon Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proc. ACM Conf. Knowledge Discovery and Data Mining*, 2012.

[150] J. O. Ramsay. Estimating smooth monotone functions. *J. Royal Statistical Society: Series B Statistical Methodology*, 60, 1998.

[151] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, 2nd edition, 2005.

[152] A. Rangarajan, H. Chui, and F. L. Bookstein. The softassign Procrustes matching algorithm. In *IPMI*, 1997.

[153] C. Rao, A. Gritai, M. Shah, and T. Fathima. View-invariant alignment and matching of video sequences. In *Proc. IEEE Int'l Conf. Computer Vision*, 2003.

[154] T. Rapcsák. On minimization on Stiefel manifolds. *European J. Operational Research*, 143(2):365–376, 2002.

[155] P. D. Ravikumar and J. D. Lafferty. Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *Proc. Int'l Conf. Machine Learning*, 2006.

[156] T. Robertson, F. T. Wright, and R. L. Dykstra. *Order restricted statistical inference*. Wiley, 1988.

[157] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996.

[158] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[159] K. Rohr. Incremental recognition of pedestrians from image sequences. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1993.

[160] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. Int'l Conf. 3D Vision*, 2001.

[161] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[162] B. Sapp, A. Toshev, and B. Taskar. Cascaded models for articulated pose estimation. In *Proc. European Conf. Computer Vision*, 2010.

[163] B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[164] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int'l J. Computer Vision*, 47(1-3):7–42, 2002.

[165] C. Schellewald and C. Schnörr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR*, 2005.

[166] C. Schellewald, S. Roth, and C. Schnörr. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In *DAGM-Symposium*, 2001.

[167] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. *Biological Sciences*, 244(1309):21–26, 1991.

[168] T. B. Sebastian, P. N. Klein, and B. B. Kimia. On aligning curves. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(1):116–125, 2003.

[169] A. Shapiro, Y. Cao, and P. Faloutsos. Style components. In *Proc. Graphics Interface*, pages 33–39, 2006.

[170] L. S. Shapiro and M. Brady. Feature-based correspondence: an eigenvector approach. *Image Vision Computing*, 10(5):283–288, 1992.

[171] S. Shariat and V. Pavlovic. Isotonic CCA for sequence alignment and activity recognition. In *Proc. IEEE Int'l Conf. Computer Vision*, 2011.

[172] J. Shotton, R. B. Girshick, A. W. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(12):

2821–2840, 2013.

[173] H. Sidenbladh, M. J. Black, and D. J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *Proc. European Conf. Computer Vision*, 2000.

[174] L. Sigal and M. J. Black. Predicting 3d people from 2d pictures. In *AMDO*, 2006.

[175] E. Simo-Serra, A. Ramisa, G. Alenyà, C. Torras, and F. Moreno-Noguer. Single image 3d human pose estimation from noisy observations. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[176] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *American Mathematical Society*, 35(2):876–879, 1964.

[177] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. IEEE Int'l Conf. Computer Vision*, 2003.

[178] Y. Suh, M. Cho, and K. M. Lee. Graph matching via sequential Monte Carlo. In *Proc. European Conf. Computer Vision*, 2012.

[179] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. F. Tappen, and C. Rother. A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2008.

[180] Y. Tian, J. Yan, H. Zhang, Y. Zhang, X. Yang, and H. Zha. On the convergence of graph matching: Graduated assignment revisited. In *Proc. European Conf. Computer Vision*, 2012.

[181] P. Tormene, T. Giorgino, S. Quaglini, and M. Stefanelli. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine*, 45(1), 2009.

[182] P. H. S. Torr. Geometric motion segmentation and model selection. *Phil. Trans. Royal Society of London A*, 356:1321–1340, 1998.

[183] P. H. S. Torr. Solving Markov random fields using semidefinite programming. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, 2003.

[184] L. Torresani, V. Kolmogorov, and C. Rother. Feature correspondence via graph matching: Models and global optimization. In *Proc. European Conf. Computer Vision*, 2008.

[185] L. Torresani, V. Kolmogorov, and C. Rother. A dual decomposition approach to feature correspondence. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(2):259–271, 2013.

[186] N. Trendafilov. On the $l_1$ Procrustes problem. *Future Generation Computer Systems*, 19 (7):1177–1186, 2004.

[187] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23:31–42, 1976.

[188] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988.

[189] R. Urtasun, D. J. Fleet, and P. Fua. 3d people tracking with Gaussian process dynamical models. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.

[190] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or. A survey on shape correspondence. *Computer Graphics Forum*, 30(6):1681–1707, 2011.

[191] B. J. van Wyk and M. A. van Wyk. A POCS-based graph matching algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(11):1526–1530, 2004.

[192] A. Veeraraghavan, A. Srivastava, A. K. Roy Chowdhury, and R. Chellappa. Rate-invariant recognition of humans and their activities. *IEEE Trans. Image Processing*, 18(6):1326–1339, 2009.

[193] J. Von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. *Contributions to the Theory of Games*, 2:5–12, 1953.

[194] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[195] M. J. Wainwright, T. Jaakkola, and A. S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. Information Theory*, 51 (11):3697–3717, 2005.

[196] C. Wang and S. Mahadevan. Manifold alignment using Procrustes analysis. In *Proc. Int'l Conf. Machine Learning*, 2008.

[197] H. Wang, M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *Proc. British Machine Vision Conference*, 2009.

[198] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *Int'l J. Computer Vision*, 103(1):60–79, 2013.

[199] J. Wang and Y. Wu. Learning maximum margin temporal warping for action recognition. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[200] O. Wang, C. Schroers, H. Zimmer, M. H. Gross, and A. Sorkine-Hornung. Videosnapping: interactive synchronization of multiple videos. *ACM Trans. Graphics*, 33(4):77, 2014.

[201] X. K. Wei and J. Chai. VideoMocap: modeling physically realistic human motion from monocular video sequences. *ACM Trans. Graphics*, 29(4), 2010.

[202] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[203] Y. Weiss and J. Pearl. Belief propagation: technical perspective. *Communications of ACM*, 53(10):94, 2010.

[204] T. Werner. A linear programming approach to max-sum problem: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(7):1165–1179, 2007.

[205] J. M. Winters and Y. Wang. Wearable sensors and telerehabilitation. *IEEE Engineering in Medicine and Biology Magazine*, 22(3):56–65, 2003.

[206] I. W. Wright and E. J. Wegman. Isotonic, convex and related splines. *Annals of Statistics*, pages 1023–1035, 1980.

[207] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. *Int'l J. Computer Vision*, 75(2): 247–266, 2007.

[208] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In

*Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011.

[209] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013.

[210] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation - an empirical study. *J. Machine Learning Research*, 7:1887–1907, 2006.

[211] A. Yao, Juergen Gall, and L. J. Van Gool. Coupled action recognition and pose estimation from multiple views. *Int'l J. Computer Vision*, 100(1):16–37, 2012.

[212] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained monocular 3d human pose estimation by action detection and cross-modality regression forest. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[213] A. L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15 (4):915–936, 2003.

[214] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

[215] M. Zaslavskiy, F. R. Bach, and J.-P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Bioinformatics*, 25(12):1259–1267, 2009.

[216] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2008.

[217] N. Zhang, R. Farrell, F. N. Iandola, and T. Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[218] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. D. Bourdev. PANDA: Pose aligned networks for deep attribute modeling. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2014.

[219] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int'l J. Computer Vision*, 13(2):119–152, 1994.

[220] Y. Zheng and D. S. Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4):643–649, 2006.

[221] F. Zhou and F. De la Torre. Canonical time warping for alignment of human behavior. In *Proc. Neural Information Processing Systems*, 2009.

[222] F. Zhou and F. De la Torre. Factorized graph matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[223] F. Zhou and F. De la Torre. Generalized time warping for alignment of human motion. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[224] F. Zhou and F. De la Torre. Deformable graph matching. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[225] F. Zhou and F. De la Torre. Spatio-temporal matching for human detection in video. In *Proc. European Conf. Computer Vision*, 2014.

[226] F. Zhou, J. Brandt, and Z. Lin. Exemplar-based graph matching for robust facial landmark localization. In *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[227] F. Zhou, F. De la Torre, and J. K. Hodgins. Hierarchical aligned cluster analysis for temporal clustering of human motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 35(3):582–596, 2013.

[228] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[229] B. Zitová and J. Flusser. Image registration methods: a survey. *Image Vision Computing*, 21(11):977–1000, 2003.