NORTHWESTERN UNIVERSITY

Collaborative Multiple Kernel Tracking: Theory and Algorithms

A THESIS

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

For the degree
MASTER OF SCIENCE

Field of Electrical and Computer Engineering

By Zhimin Fan
Supervising Professor: Ying Wu

Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois 60208, USA

December, 2005

Abstract

To make the kernel-based tracking algorithms more reliable, in this work, we mainly deal with two major singular cases in kernel-based tracking, concerned with kernel observability and tracking stability.

Singular kernel observability indicates that the motions of interest cannot be uniquely recovered by the kernel. We present a novel multiple collaborative kernel approach, in which a complex motion is represented by a set of inter-correlated simpler motions. With this formulation, we present a rigorous analysis on a critical issue of kernel observability and obtain a criterion, based on which we propose a new method using collaborative kernels that has the theoretical guarantee of enhanced observability. This new method has been shown to be computationally efficient in both theory and practice, which can be readily applied to complex motions such as articulated motions.

Another singular case, unstableness in tracking, is caused by inappropriate kernel placement, which requires research on optimal kernel placement. The theoretical analysis presented in this work indicates that the optimal kernel placement can be evaluated based on a closed-form criterion, and achieved efficiently by a novel gradient-based algorithm. Based on that, new methods for temporal-stable multiple kernel placement and scale-invariant kernel placement are also proposed. These new theoretical results and new algorithms greatly advance the study of kernel-based tracking in both theory and practice. Extensive experimental results demonstrate the improved tracking reliability.

Acknowledgments

I would like to thank my advisor, Prof. Ying Wu, for his invaluable guidance, enlightening advice, endless encouragement and support throughout my study here in Northwestern University. The fruitful discussions with Prof. Ying Wu not only help me to go through many difficulties, but also add more deep insights into this research work.

I would like to thank Prof. Ying Wu, Prof. Aggelos Katsaggelos, and Prof. Thrasos Pappas to kindly be my committee members, and thank Prof. Allen Taflove for his support and help. I would like to thank to my lab mates, Gang Hua, Ting Yu, Ming Yang, Shengyang Dai, and Junsong Yuan for their selfless efforts and cooperations to make the study and living here enjoyable. I would also like to thank the members of IVPL for their help.

Finally, I will thank my parents and my girlfriend Na, for their love, understanding, consistent support and encouragement, without which this work will not become true.

This work was supported in part by National Science Foundation (NSF) Grant IIS-0308222, NSF IIS-0347877 (CAREER), Northwestern startup funds and the Murphy Fellowships.

Contents

1 Introduction **12**

2 Related Work **19**

2.1 Kernel-based Tracking 19

2.1.1 Mean shift analysis 19

2.1.2 Mean shift tracking 22

2.1.3 Closed-form tracking 25

2.2 Singularities in Kernel-based Tracking 27

2.2.1 Kernel-observability and improvement on kernel design . 28

2.2.2 Tracking stability and optimal kernel placement 30

3 Multiple Collaborative Kernel Tracking **32**

3.1 Kernel-observability Analysis 32

3.1.1 Example 1: a single kernel 36

3.1.2 Example 2: kernel concatenation 37

3.1.3 Example 3: kernel combination 38

3.2 Multiple Collaborative Kernels 41

3.2.1 Enhancing the observability 42

3.2.2 The system description 47

3.2.3 The collaboration 52

3.3 Experiments 55

3.3.1 Tracking structured object 55

3.3.2 Tracking articulated objects 59

3.3.3 Using the subspace model 60

3.4 Remarks 66

4 Efficient Optimal Kernel Placement for Reliable Visual Tracking 67

4.1 Optimal Single Kernel Placement 67

4.1.1 Applying the condition theory 68

4.1.2 Interpretation of the condition number criterion using the 2-norm 71

4.1.3 An equivalent condition number 72

4.1.4 Find optimal kernel placement efficiently 77

4.2 Multiple Kernel Placement 81

4.3 Scale-invariant Kernel Placement 84

4.3.1 Scale selection 86

4.3.2 Pruning text regions 88

4.4 Discussions 89

4.4.1 Region selection vs. feature point selection 89

4.4.2 Interpretation of condition number criterion using the S-norm 91

4.5 Experiment 92

4.5.1 Single kernel 93

4.5.2 Multiple collaborative kernels 94

4.6 Remarks 98

5 Conclusions 100

List of Tables

List of Figures

3.1.1 A comparison of single kernel (top row), kernel concatenation (middle row), and kernel combination (bottom row). 40

3.1.2 The surfaces of $1 - \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}}\|^2$ of (a) single kernel, (b) kernel concatenation, and (c) kernel combination. 41

3.2.1 The length constraint on a rod. 42

3.3.1 Tracking a handset. 56

3.3.2 Tracking a rod-shaped bottle. 57

3.3.3 Tracking a finger. 58

3.3.4 Tracking the articulated body with two arms. 59

3.3.5 Tracking an articulated structure. 60

3.3.6 Labelled interest points for subspace model learning. 61

3.3.7 Tracking three parts of interest on a box. 62

3.3.8 Error comparison of single and collaborative kernels against the ground truth. 63

3.3.9 Tracking four parts of interest on a box. 64

3.3.10 Tracking the head and shoulders. 64

3.3.11 Tracking a magazine cover with deformation. 65

4.1.1 Synthesized image patterns and their κ_2, κ_S. 77

4.1.2 column (a): yellow: image region, red: kernel. column (b), (c), and (d) are κ_2, κ_S and $\kappa_S - \kappa_2$ evaluated in the region of column (a), respectively. 78

4.1.3 column (a): the red rectangle indicates the start kernel position, the green one is the optimized kernel placement found by the gradient-based searching algorithm. column (b): the corresponding descending value of the κ_S. column (c): the regions covered by the kernel, which is moving along the direction of the gradient towards the good placement. The red line with a spot indicates the center of mass of each color component. 82

4.3.1 In each figure (a)-(h). Top row: scale-invariant kernel placement. Larger circle means higher density of converged kernel samples. Bottom row: good kernel regions with appropriate scale selection after pruning texture regions. 86

4.5.1 Tracking with (bottom row) and without (top row) kernel placement optimization. 94

4.5.2 Tracking with (bottom row) and without(top row) kernel placement optimization. 95

4.5.3 Multiple kernel tracking, with (bottom row) and without (top row) placement optimization. 96

4.5.4 Multiple kernel tracking, with (bottom row) and without (top row) collaboration. 97

4.5.5 Multiple kernel tracking, with (bottom row) and without (top row) collaboration. 98

4.5.6 Multiple kernel tracking, with (bottom row) and without (top row) collaboration. 99

Chapter 1

Introduction

Kernel-based methods [6, 30] have attracted much attention in computer vision [8, 10, 14, 15, 31] and have recently shown promising performance in the challenging problem of visual tracking [9]. In this context, the representation of the object being tracked is the convolution of the object features with a spatially weighted kernel, which enables efficient gradient based optimization methods, such as mean shift [8] or closed-form method [17], to search for the best match to the target model based on the collected visual measurements (or observations). Thus, one of the most appealing merits of kernel-based trackers is their low computational cost, compared with other commonly employed tracking schemes, such as particle filters [22] or exhaustive template matching.

Since the kernel-based tracking methods are gradient-based differential approaches, their performances are largely affected by the quality of the searching directions calculated from the measurements (i.e., the discrepancy between the candidates and the target model). However, in practice, singularities are often observed in computing the *gradient*, which may greatly impair the tracking performance. In this work, we address the two major singularities, propose algorithms associated with proved theories, which can help to achieve a much more reliable tracking performance.

One kind of singularity is about the *kernel observability*. That is, the searching direction is indifferent to the measurements, i.e, the measurements become more or less invariant to some motion parameters, such that these motion parameters are not *uniquely* recoverable or observable, indicating a deficient kernel observability. Regarding this singular case, three critical issues of both theoretical and practical importance need to be investigated:

- Is there a criterion or a test that detects such singularities and checks the observability of the motion?
- Is there a principled way of kernel design to prevent or alleviate such singularities?

- Can we cope with such singularities in more complex motions (e.g., articulation) while still achieving computational efficiency?

There have been some initial studies related to these questions. For example, in [7], to deal with the problem that most kernels, being scale-invariant, cannot recover the scale changes of the target, a method was proposed to combine multiple kernels of different resolutions. An outstanding initial investigation on multiple kernels was presented in [17], where an unconstrained linear least square formulation was given and the motion singularity can be revealed by the rank deficiency when approaching its solution, based on which a multiple kernel method was proposed to possibly reduce the risk of rank deficiency.

These initial investigations on multiple kernels are meaningful, but they are inadequate. For example, although several suggestions have been made in [17] on designing multiple kernels, it is desirable to have a more rigorous theoretical guarantee on motion recoverability and a more principled and generalizable approach to kernel design. In addition, complex motions (e.g., motions of articulated bodies) pose a great challenge to most existing kernel-based tracking algorithms which are largely confined by single target and simple motions, and this is a topic remained largely unexplored among the literatures of kernel-based methods. Although many top-down algorithms have been explored for complex motion [4, 33], they are in general computationally demanding. Thus, it will be very meaningful if the bottom-up kernel-based solutions can be found.

Inspired by [17, 33], we present a novel *multiple collaborative kernel* approach to visual tracking. This approach treats kernel-based tracking in a more general formulation, i.e., a relaxation and constraints formulation, in which a complex motion can be represented by a set of inter-correlated simpler motions. In this new formulation, the state equation describes the constraints among these simpler motions, and the measurement equation characterizes the independent visual measurement processes of these simpler motions. With this formulation, our work present a rigorous theoretical analysis on the singularity issue, i.e., kernel observability, and presents the observability criterion. Based on this, we propose the multiple collaborative kernel method that has the theoretical guarantee of enhanced observability. This new method has been shown to be computationally efficient in both theory and practice.

The proposed design of "multiple collaborative kernels" closely follows the theoretical concerns on "kernel-observability", i.e., singularity in motion detection, and substantially broadens the applicability of kernel based methods for tracking of multiple targets with complex motions, such as the articulated body motions.

Besides the singular case of *kernel observability*, another kind of major singularity is about the *tracking stability*. All the representative kernel-based tracking methods [7, 9, 13, 17] assume that unique and stable motion estimation can be obtained as in the well-conditioned cases. In other words, a small perturbation of the placement of the kernel does not change much the motion estimation. Unfortunately, evidence from the practice challenges this assumption. For example, in mean shift tracking, it is often observed that different initializations of the tracker (i.e., delineate the region to track and place the kernel accordingly) may largely influence the performance. If we put the same kernel at one place, the tracker may work well; but when choosing a slightly different place, the tracker may fail unexpectedly, e.g., even a small perturbation can change the estimated motion significantly, thus bringing unstableness into the tracking. This raises another interesting and critical question: **is there an optimal placement for the kernels to achieve reliable tracking?** Specifically:

- How can we evaluate the sensitivity of a placement?
- Does there exist a computationally efficient way to find the optimal kernel placement?

- How can we place multiple kernels if a training sequence is available?
- Does there exist a scale-invariant kernel placement?

In this work, we also present our study in search of the answers to the above intriguing questions in order to achieve more reliable tracking results. Our study starts with a conjecture that subregions of the target may play different roles in tracking, since some subregions of the target may be more reliable for tracking while others may not. We provide a detailed analysis in order to identify those regions, and derive a *closed-form* criterion for evaluating the sensitivity of kernel placement. To make the optimal kernel placement feasible, we derive a *gradient-based* algorithm to efficiently search for an optimal placement, which greatly reduces the computational cost compared with a brute force way of examining all the possible placement on the image exhaustively. We also propose a method to discover temporal-stable kernels for multiple kernel placement, and study the issue of scale-invariant kernel placement.

Advancing the state of the art, the contributions of this work include:

(1) the theoretical results that unify the study of the motion observability issue in most kernel-based methods including single and multiple kernels;

(2) a principled way of designing observable kernels, i.e.. the multiple col-

laborative kernels, that can be easily generalized to complex objects and motions;

(3) an efficient computational paradigm to cope with complex objects and motions due to the “collaboration” among a set of inter-correlated kernels, each of which only takes charge of recovering a simpler motion;

(4) a closed-form criterion for choosing the optimal kernel placement, on which a much more reliable tracking performance can be achieved;

(5) a gradient-based searching algorithm to find such optimal kernel placements, which greatly reduces the computational cost compared with the commonly used exhaustive searching.

The remainder of the thesis is organized as follows. Chapter 2 introduces the related work and the two major singularities in kernel-based tracking. Chapter 3 presents our detailed analysis on multiple collaborative kernel tracking, which deals with the singular case of kernel observability. Chapter 4 presents our study on the issue of optimal kernel placement, which deals with the singular case of tracking stability. Conclusions are made in chapter 5.

Chapter 2

Related Work

2.1 Kernel-based Tracking

2.1.1 Mean shift analysis

Mean shift analysis is mainly used for exploring the maximal or minimal properties in density estimation. Given a set $\{\mathbf{x}_i\}, i = 1, \ldots, n$ of n points in the d-dimensional space $\mathcal{R}^d$, the *multivariate kernel density estimate* with kernel $K(\mathbf{x})$ and window radius (band-width) h, computed in the point $\mathbf{x}$ is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{x_i}}{h}\right) \qquad (2.1.1)$$

where, $K(\mathbf{x})$ can be the multivariate Epanechnikov kernel

$$K_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\|\mathbf{x}\|^2) & \text{if } \|\mathbf{x}\| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.1.2}$$

and c_d is the volume of the unit d-dimensional sphere. Another commonly used kernel is the multivariate normal

$$K_N(\mathbf{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \tag{2.1.3}$$

Denote the *profile* of a kernel K as a function k such that $K(\mathbf{x}) = k(\|\mathbf{x}\|^2)$. For example, the Epanechnikov profile is

$$k_E(\mathbf{x}) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-x) & \text{if } x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.1.4}$$

and the normal profile is given by

$$k_N(\mathbf{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}x\right) \tag{2.1.5}$$

So, the density estimate can be written as

$$\hat{f}_K(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} k\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right) \tag{2.1.6}$$

By denoting $g(x) = -k'(x)$, the derivative of $\hat{f}_K(\mathbf{x})$ is computed as follows,

$$\begin{aligned}\hat{\nabla} f_K(\mathbf{x}) \equiv \nabla \hat{f}_K(\mathbf{x}) &= \tfrac{2}{nh^{d+2}} \textstyle\sum_{i=1}^{n}(\mathbf{x}-\mathbf{x}_i)k'\left(\|\tfrac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right) \\ &= \tfrac{2}{nh^{d+2}} \textstyle\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{x})g\left(\|\tfrac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right) \\ &= \tfrac{2}{nh^{d+2}} \left[\textstyle\sum_{i=1}^{n} g\left(\|\tfrac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)\right]\left[\tfrac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x}\right]\end{aligned} \tag{2.1.7}$$

Note that the derivative of the Epanechnikov profile is the uniform profile, while the derivative of the normal profile remains a normal.

The last bracket in Eq.(2.1.7) contains the sample mean shift vector

$$M_{h,K}(\mathbf{x}) \equiv \frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{x} \tag{2.1.8}$$

The *mean shift procedure* is defined recursively by computing the mean shift vector $M_{h,K}(\mathbf{x})$ and translating the center of kernel by $M_{h,K}(\mathbf{x})$.

Let $\mathbf{y}_j, j = 1, 2, ...$ represnet the sequence of successive locations of kernel K, where

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{y}_j-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n} g\left(\|\frac{\mathbf{y}_j-\mathbf{x}_i}{h}\|^2\right)} \tag{2.1.9}$$

is the weighted mean at $\mathbf{y}_j$ computed with kernel K and $\mathbf{y}_1$ is the center of the initial kernel. The corresponding density computed with kernel K in the points Eq.(2.1.9) are

$$\hat{f}_K = \left\{ \hat{f}_K(j) \right\}_{j=1,2,\ldots} \equiv \left\{ \hat{f}_K(\mathbf{y}_j) \right\}_{j=1,2,\ldots} \qquad (2.1.10)$$

A theorem is proved in [9], which states that

Theorem 1 *If the kernel K has a convex and monotonic decreasing profile, the sequences Eq.(2.1.9) and (2.1.10) are convergent.*

According to Eq.(2.1.7), the $\mathbf{y}_{j+1}$ in Eq.(2.1.9) is actually the point, which yields *zero* in the derivative of density estimated at point $\mathbf{y}_j$. Therefore, the density estimated at $\mathbf{y}_{j+1}$ is a local optimum in the neighborhood of $\mathbf{y}_j$. The convergency of the point sequence of Eq.(2.1.9) and the density estimation sequence of Eq.(2.1.10) actually tells that the local extremity of density can be found by the mean shift iteration.

2.1.2 Mean shift tracking

The above mean shift analysis can be used for efficient object tracking [9]. In this subsection, the notations mostly follow that in [9].

Assume $\{\mathbf{x_i}\}_{\mathbf{i=1\ldots n}}$ be the pixel locations of the target. For each pixel $\mathbf{x}_i$, a binning function $b(\mathbf{x_i})$ maps a predefined feature, e.g., the color, of $\mathbf{x}_i$ onto a histogram bin u, with $u \in \{1\ldots m\}$. Let K be a spatially weighted kernel.

Then, a histogram representation of the target **q**$= [q_1, q_2, \ldots, q_m]^T \in \mathbb{R}^m$ can be computed as,

$$q_u = \frac{1}{C} \sum_{i=1}^{n} K(\mathbf{x}_i - \mathbf{c}) \delta(b(\mathbf{x}_i), u), \tag{2.1.11}$$

where δ is the Kronecker delta function, **c** is the kernel center and C is the normalization factor. Using a decaying kernel K actually means that the pixels centered at the center of the kernel contribute more to the color histogram, while peripheral pixels are the least reliable.

Given an initial start at location $\mathbf{c}_0$, the core problem in tracking is to find a best displacement $\Delta\mathbf{c}$ such that the measurement $\mathbf{p}(\mathbf{c}_0 + \Delta\mathbf{c})$ at the new location best matches the target $\mathbf{q}$, i.e.,

$$\Delta\mathbf{c}^* = \arg\min_{\Delta c} O(\mathbf{q}, \mathbf{p}(\mathbf{c}_0 + \Delta\mathbf{c})), \tag{2.1.12}$$

where $O(\cdot,\cdot)$ is the objective function for matching. For example, it can be the Bhattacharyya coefficient [9]:

$$O_B(\Delta\mathbf{c}) \overset{\triangle}{=} -\langle \sqrt{\mathbf{q}}, \sqrt{\mathbf{p}(\mathbf{c}_0 + \Delta\mathbf{c})} \rangle = -\sqrt{\mathbf{q}^T}\sqrt{\mathbf{p}(\mathbf{c}_0 + \Delta\mathbf{c})}.$$

Denote $\mathbf{c} = \mathbf{c}_0 + \Delta\mathbf{c}$, we can approximate $\sqrt{\mathbf{q}^T}\sqrt{\mathbf{p}(\mathbf{c})}$ by using Taylor series expansion,

$$\sqrt{\mathbf{q}^T}\sqrt{\mathbf{p}(\mathbf{c})} \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(\mathbf{c}_0)q_u} + \frac{1}{2}\sum_{u=1}^{m}p_u(\mathbf{c})\sqrt{\frac{q_u}{p_u(\mathbf{c}_0)}} \tag{2.1.13}$$

Then, introducing Eq.(2.1.11) in Eq.(2.1.13) we obtain

$$\sqrt{\mathbf{q}^T}\sqrt{\mathbf{p}(\mathbf{c})} \approx \frac{1}{2}\sum_{u=1}^{m}\sqrt{p_u(\mathbf{c}_0)q_u} + \frac{1}{2C}\sum_{u=1}^{n}w_i k\left(\|\frac{\mathbf{c}-\mathbf{x}_i}{h}\|^2\right) \tag{2.1.14}$$

where

$$w_i = \sum_{u=1}^{m}\delta(b(\mathbf{x}_i),u)\sqrt{\frac{q_u}{p_u(\mathbf{c}_0)}} \tag{2.1.15}$$

In Eq.(2.1.14), since the first term is independent of **c**, and the second term represents the *density estimate* computed with kernel profile k at **c**, with the data being weighted by w_i. Therefore, the maximization of this density,the minimization of the objective function $O_B(\Delta\mathbf{c})$, can be efficiently achieved based on the mean shift iteration, i.e., given the current center of kernel $\mathbf{c}_0$, the direction to search for maximum is along the mean shift vector,

$$\frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\|\frac{\mathbf{c}_0-\mathbf{x}_i}{h}\|^2\right)}{\sum_{i=1}^{n}g\left(\|\frac{\mathbf{c}_0-\mathbf{x}_i}{h}\|^2\right)} - \mathbf{c}_0$$

since this direction is just the *gradient* of the current density estimate.

2.1.3 Closed-form tracking

A more concise matrix form of the object histogram, Eq.(2.1.11), can be written as [17]:

$$\mathbf{q}(\mathbf{c}) = \mathbf{U}^T \mathbf{K}(\mathbf{c}), \tag{2.1.16}$$

where

$$\mathbf{U} = \begin{bmatrix} \delta(b(\mathbf{x}_1), u_1) & \dots & \delta(b(\mathbf{x}_1), u_m) \\ \vdots & \vdots & \vdots \\ \delta(b(\mathbf{x}_n), u_1) & \dots & \delta(b(\mathbf{x}_n), u_m) \end{bmatrix} \in \mathbb{R}^{n \times m},$$

and

$$\mathbf{K} = \frac{1}{C} \begin{bmatrix} K(\mathbf{x}_1 - \mathbf{c}) \\ \vdots \\ K(\mathbf{x}_n - \mathbf{c}) \end{bmatrix} \in \mathbb{R}^n.$$

This expression facilitates a closed-form method for object tracking.

In general, for the target model, the kernel is centered at $\mathbf{0}$, and we denote it by $\mathbf{q} = \mathbf{U}^T \mathbf{K}$. By the same token, we can represent the histogram observed at a given candidate region centered at **c** as:

$$\mathbf{p}(\mathbf{c}) = \mathbf{U}^T \mathbf{K}(\mathbf{c}). \tag{2.1.17}$$

The Matusita metric, which is equivalent to the Bhattacharyya coefficient, is used in [17] as the objective function, which is:

$$O_M(\Delta\mathbf{c}) \stackrel{\triangle}{=} \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c}+\Delta\mathbf{c})}\|^2. \tag{2.1.18}$$

Linearizing Eq.(2.1.18), we obtain

$$\mathbf{M}\Delta\mathbf{c} = \sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})},$$

where $\sqrt{\mathbf{q}}$, $\sqrt{\mathbf{p}(\mathbf{c})} \in \mathbb{R}^m$, $\Delta\mathbf{c} \in \mathbb{R}^r$, $\mathbf{M} \in \mathbb{R}^{m\times r}$,

$$\begin{aligned} \mathbf{M} &= \tfrac{1}{2}\mathtt{diag}(\mathbf{p}(\mathbf{c}))^{-\frac{1}{2}}\mathbf{U}^T\mathbf{J_K}(\mathbf{c}), \\ \mathbf{J_K}(\mathbf{c}) &= \begin{bmatrix} \nabla_c K(\mathbf{x}_1 - \mathbf{c}) \\ \nabla_c K(\mathbf{x}_2 - \mathbf{c}) \\ \vdots \\ \nabla_c K(\mathbf{x}_n - \mathbf{c}) \end{bmatrix}, \end{aligned} \tag{2.1.19}$$

and $\mathtt{diag}(\mathbf{p})$ represents the matrix with $\mathbf{p}$ on its diagonal, r is the dimensionality of the motion parameters.

Thus, the solution $\Delta\mathbf{c}$, which minimizes the difference between $\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$, can be computed in closed-form by solving the above linear equation as

$$\Delta\mathbf{c} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T(\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}).$$

2.2 Singularities in Kernel-based Tracking

Since the kernel-based tracking methods are essentially gradient-based differential approaches, their performances are greatly influenced by the quality of the searching directions calculated based on the local measurements. In practice, great deteriorations can be observed during tracking, because there exist singular cases in computing the *gradient*.

One kind of singularity is about the *kernel observability*. That is, we sometimes are plagued in the singular situation where the same optimal value of $O(\Delta\mathbf{c})$ can be achieved over a continuous range, i.e., any candidate region induced by the movement in this range matches the target equally well. In other words, the motion parameters can not be uniquely determined, or can not be fully *observed* through the kernel. This kind of singularity is due to the inadequate kernel design, which impairs the kernel's observability to the underlying object motions. We will further investigate into this issue and present our solution in Chapter 3.

Another kind of singularity is about the *tracking stability*. When we put the kernel into some places, it is possible that even a small perturbation can change the estimated motion significantly, thus bringing unstableness into the tracking. To address this issue, we present our study in Chapter 4, where a detailed analysis is given to identify the good regions to track, a closed-form criterion is derived for evaluating the sensitivity of kernel placement and for choosing the *optimal* kernel placement. To make the optimal kernel placement feasible, we also propose a *gradient-based* algorithm to efficiently search for such a placement, which greatly reduces the computational cost compared with the exhaustive search.

2.2.1 Kernel-observability and improvement on kernel design

Kernels are considered as measuring tools to obtain *observations* of the feature space, which can guide optimization algorithms, often gradient based approaches, to solve a certain problem. Here, the *design of the kernel* plays a key role. Inadequate kernel design will impair the kernel's observability to the underlying parameters needed to be estimated, which will mislead the optimization procedure, and results in very poor performance. The recent awareness of the inadequate kernel design calls for more appropriate schemes to construct kernels.

One type of the inadequate kernel design is about the resolution of the kernel's sensitivity. When exploring the color space for image segmentation [8], in order to make the kernels being able to acquire a more precise and pertinent measurement, kernels having various bandwidths [10, 29] or shapes [31] have been proposed.

Another type of the inadequate kernel design is about the lack of the measuring tools, i.e., kernels associated with a certain property in the feature space are missing. In the kernel-based tracking problem [9], traditional kernels cannot react to the scale changes of the target. A set of kernels in the scale space is thus constructed in [7] to account for this limitation.

An interesting inadequate kernel design encountered recently in the kernel-based tracking problem poses new challenges to the issue of kernel-observability: whether or not the underlying object motions can be *uniquely determined* from the kernel observation? This is also the main topic of our study in Chapter 3. Pointed out in the initial work by Hager *et.al.* [17], the measurement obtained from the kernels with some deficiency is insensitive to certain object motions and thus will induce singularities in the "unique recovery" of those motion parameters. More intuitively, some kernels, by construction, are blind to some certain object motions, which will lead to tracking failures. Multiple kernels are then used in [17] to increase the measurement space, which is supposed to accommodate those previously unobservable motions.

In this thesis, we give an in-depth and more rigorous investigation into the issue of "kernel-observability". A criterion is derived from a more general description of the tracking problems, which not only supervises the "unique recovery" of the object motion, but also brings out a *collaborative kernel tracking scheme*. This scheme offers substantial advantages over the previous single independent kernel method.

2.2.2 Tracking stability and optimal kernel placement

Besides the issue of kernel-observability, another critical issue in tracking is the stability. Significant changes in kernel positions caused by small perturbations from observation is an undesirable phenomena of *unstableness* in tracking. Different kernel placements have different stabilities in tracking. We will present our study on analyzing the stability, or in other words, sensitivity, in choosing kernel placement and propose the criterion to select regions, which are *optimal* by the construction of the kernel-based tracker.

To the best our knowledge, the question that "what is the *optimal region* to place a kernel for tracking?" has remained largely unexplored among the literatures. In [5][24][25][28], the problem of selecting good *feature point* for tracking has been studied based on eigenvalue analysis. Some work also extends the point matching framework to address other geometrical features such as lines [20][32]. But, the representations of good feature points/lines are largely different from that of good regions for tracking, so is the analytical result.

Our work on exploring optimal kernel placement is new. Besides, we also propose a method to discover temporal-stable kernels for multiple kernel placement, and study the issue of scale-invariant kernel placement.

Chapter 3

Multiple Collaborative Kernel Tracking

3.1 Kernel-observability Analysis

The issue of "kernel-observability" mentioned in the previous chapter can be related to the "system-observability" of a more general system in Eq.(3.1.1) for a better definition and explanation. We omit the noise terms for clarity, since it does not affect the analysis.

$$\left\{\begin{array}{lcl} \Omega(\mathbf{x}) & = & 0 \\ \mathbf{y} & = & \mathcal{M}(\mathbf{x}), \end{array}\right. \tag{3.1.1}$$

where $\Omega(\mathbf{x})$ represents the inherent property of the state variable $\mathbf{x}$, such as the complexity, self-contained constraint or the system dynamics, and $\mathcal{M}$ denotes the observation or measurement process. In this system, the state variable $\mathbf{x}$ is hidden and can only be estimated through the measurement $\mathbf{y}$. In the tracking scenario, the state variable refers to the motion to be estimated. Here we do not limit our discussions only to the 2D displacements, but generalize it to r dimensional motion vector, i.e., $\mathbf{x} \in \mathbb{R}^r$. A critical issue is whether or not $\mathbf{x}$ can be *uniquely determined* from $\mathbf{y}$, i.e., the *observability* of this system.

In the context of kernel tracking, we treat

$$\mathbf{x} \overset{\triangle}{=} \Delta\mathbf{c}.$$

Our analysis is based on the linearization of the system at a given initial start $\mathbf{c}$, since the local property of $\mathbf{c}$ is the mostly concerned in the tracking problem. The collected image evidence for $\mathbf{c} + \Delta\mathbf{c}$ is the difference between the target and the candidate, i.e., $\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c} + \Delta\mathbf{c})}$. Linearizing it w.r.t. $\Delta\mathbf{c}$, we have

$$\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})} = \mathbf{M}\Delta\mathbf{c},$$

where $\sqrt{\mathbf{q}}$, $\sqrt{\mathbf{p}(\mathbf{c})} \in \mathbb{R}^m$, $\Delta\mathbf{c} \in \mathbb{R}^r$, $\mathbf{M} \in \mathbb{R}^{m\times r}$,

$$\mathbf{M} = \tfrac{1}{2}\mathtt{diag}(\mathbf{p}(\mathbf{c}))^{-\frac{1}{2}}\mathbf{U}^T\mathbf{J}_\mathbf{K}(\mathbf{c}),$$

$$\mathbf{J}_\mathbf{K}(\mathbf{c}) = \begin{bmatrix} \nabla_c K(\mathbf{x}_1 - \mathbf{c}) \\ \nabla_c K(\mathbf{x}_2 - \mathbf{c}) \\ \vdots \\ \nabla_c K(\mathbf{x}_n - \mathbf{c}) \end{bmatrix},$$

and $\mathtt{diag}(\mathbf{p})$ represents the matrix with $\mathbf{p}$ on its diagonal. This result was actually obtained in [17]. In view of this, we treat the measurement $\mathbf{y} \triangleq \sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$, and thus the linearized **measurement equation** can be written as:

$$\mathbf{y} = \mathbf{M}\Delta\mathbf{c} = \mathbf{M}\mathbf{x}. \tag{3.1.2}$$

When the motion constraints holds at $\mathbf{c} + \Delta\mathbf{c}$, i.e., $\Omega(\mathbf{c} + \Delta\mathbf{c}) = 0$, we can always linearize it as

$$\Omega(\mathbf{c}) + \Omega'(\mathbf{c})\Delta\mathbf{c} = 0.$$

Thus, when we define $l \triangleq -\Omega(\mathbf{c})$, and $\mathbf{G} \triangleq \Omega'(\mathbf{c})$, we have a linearized **system state equation**, or the **state constraint equation**:

$$l = \Omega'(\mathbf{c})\Delta\mathbf{c} = \mathbf{G}\mathbf{x}, \tag{3.1.3}$$

where $\mathbf{x} \in \mathbb{R}^r$ and $\mathbf{G} \in \mathbb{R}^{s\times r}$, s is the number of linear constraints. We have the following theorem that stipulates the kernel observability,

Theorem 2 Kernel-Observability

The system described by Eq.(3.1.2) and Eq.(3.1.3) is observable, i.e., unique recovery of **x** *is guaranteed, iff*

$$\texttt{rank}(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G}) = r, \quad \forall\gamma > 0 \tag{3.1.4}$$

i.e., $(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})$ *is of full rank.*

Proof:

Given the system state equation Eq.(3.1.3) and the measurement equation Eq.(3.1.2), we form an objective function that penalizes the measurement mismatch and the deviation from the system constraints:

$$L(\mathbf{x}) \overset{\triangle}{=} \|\mathbf{M}\mathbf{x} - \mathbf{y}\|^2 + \gamma\|\mathbf{G}\mathbf{x} - l\|^2,$$

where $\gamma > 0$. Setting the derivative to zero, we have:

$$\mathbf{x}^* = (\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})^{-1}(\mathbf{M}^T\mathbf{y} + \gamma\mathbf{G}^T l).$$

This is equivalent to the least square solution to the following system:

$$\begin{bmatrix} \mathbf{M} \\ \sqrt{\gamma}\mathbf{G} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{y} \\ \sqrt{\gamma}l \end{bmatrix}.$$

Thus the solution is unique iff the rank of $\mathbf{M}^T\mathbf{M}+\gamma\mathbf{G}^T\mathbf{G}$ is full, or $\begin{bmatrix}\mathbf{M}\\ \sqrt{\gamma}\mathbf{G}\end{bmatrix}$ has full column rank. ■

Based on this theorem, we demonstrate three examples on the unique recovery of **x**, i.e., $\Delta\mathbf{c}$, from the above system, and motivate our proposed approach of multiple collaborative kernels in Section 3.2.

3.1.1 Example 1: a single kernel

As a special case, if we do not consider the system state equation, which means the contribution of **G** vanishes, i.e., $\mathbf{G} = 0$, the observability of a single kernel, based on the Theorem 2, is given by checking $\texttt{rank}(\mathbf{M}^T\mathbf{M})$, or $\texttt{rank}(\mathbf{M})$[1].

This conclusion coincides with the SSD-based analysis in [17], where a least square problem is formulated:

$$\min_{\Delta c} \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})} - \frac{1}{2}\mathtt{d}(\mathbf{p}(\mathbf{c}))^{-\frac{1}{2}}\mathbf{U}^T\mathbf{J}_{\mathbf{K}}(\mathbf{c})\Delta\mathbf{c}\|^2.$$

Hager *et.al.* [17] pointed out the rank deficiency of $\mathbf{M} = \frac{1}{2}\mathtt{d}(\mathbf{p})^{-\frac{1}{2}}\mathbf{U}^T\mathbf{J}_{\mathbf{K}}(\mathbf{c})$ will not allow a unique solution to $\Delta\mathbf{c}$.

In order to recover $\Delta\mathbf{c}$ in this system, before taking effort to make **M** full rank,

[1]For matrix **H**, $\texttt{rank}(\mathbf{H}^T\mathbf{H}) = \texttt{rank}(\mathbf{H})$

it should be noted that $\mathtt{d}(\mathbf{p})^{-\frac{1}{2}}$ and $\mathbf{U}$ would not be rank deficient as long as the number of the non-zero values in the histogram is no less than the number of the parameters to be estimated, which is solely determined by the image and the target property.

Thus, the point that the observability gain can found its place is to change the kernel related $\mathbf{J}_{\mathbf{K}}(\mathbf{c})$, i.e., to change the ways of extracting the representative information from the objects, which motivates the methods of using multiple kernels. Two examples will be given in Sec 3.1.2 and Sec. 3.1.3, and our proposed method in Sec. 3.2.

3.1.2 Example 2: kernel concatenation

We can concatenate multiple kernels to increase the dimensionality of the measurement (i.e., the histogram). Suppose there are w kernels, each of them produces a histogram measure for the object, $\mathbf{p}_i(\mathbf{c}) = \mathbf{U}^T\mathbf{K}_i(\mathbf{c})$, where $i = 1, \ldots, w$. By vertically stacking these histograms into $\overline{\mathbf{p}}$ and $\overline{\mathbf{q}}$, it is easy to show that based on the Theorem 2, the observability is given by checking $\texttt{rank}(\mathbf{M}^T\mathbf{M})$, where

$$\mathbf{M} = \frac{1}{2}\mathtt{d}(\overline{\mathbf{p}})^{-\frac{1}{2}} \begin{bmatrix} \mathbf{U}^T & & \\ & \ddots & \\ & & \mathbf{U}^T \end{bmatrix} \begin{bmatrix} \mathbf{J}_{\mathbf{K}_1} \\ \vdots \\ \mathbf{J}_{\mathbf{K}_w} \end{bmatrix}, \qquad (3.1.5)$$

which may hopefully have full column rank to enable a unique solution to $\Delta\mathbf{c}$. This makes sense since more features have been used. This is actually the multiple kernel method suggested in [17]. In fact, the kernel concatenation implies the optimization problem as:

$$\min_{\Delta c} \sum_{i=1}^{w} \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}_i(\mathbf{c} + \Delta\mathbf{c})}\|^2.$$

3.1.3 Example 3: kernel combination

Besides kernel concatenation in Sec. 3.1.2 that uses more features, another feasible solution is kernel combination to produce new features by aggregating the histogram vectors from multiple kernels (with normalization):

$$\overline{\mathbf{q}} = \sum_{i=1}^{w} \mathbf{U}^T\mathbf{K}_i, \quad \overline{\mathbf{p}} = \sum_{i=1}^{w} \mathbf{U}^T\mathbf{K}_i(\mathbf{c}).$$

Then the measurement equation is written as:

$$\sqrt{\overline{\mathbf{q}}} - \sqrt{\overline{\mathbf{p}}(\mathbf{c})} = \mathbf{M}\Delta\mathbf{c},$$

where

$$\mathbf{M} = \frac{1}{2}\mathtt{d}(\overline{\mathbf{p}})^{-\frac{1}{2}}\mathbf{U}^T \sum_{i=1}^{w} \mathbf{J}_{\mathbf{K}_i}. \tag{3.1.6}$$

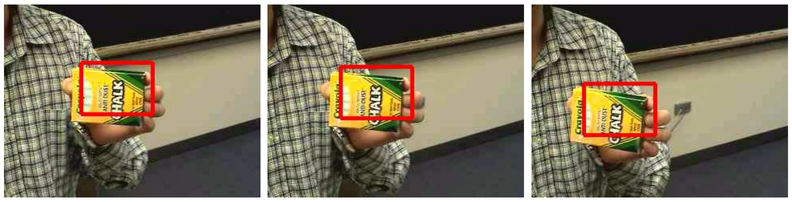
This may also make the matrix $\mathbf{M}^T\mathbf{M}$ full rank. In essence, as long as the measurement matrix $\mathbf{M}$ can well depict the characteristics around $\mathbf{c}$, we can find a proper $\Delta\mathbf{c}$ in the neighborhood that minimizes $\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c} + \Delta\mathbf{c})}$.

Here, we give an illustrative example. For comparison, we employ the same roof kernels as in [17], with length l, span s, center $\mathbf{c}$ and normal vector $\mathbf{n}$.

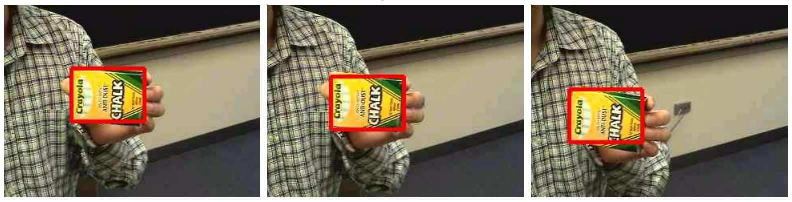
$$K_{roof}(\mathbf{x}; \mathbf{c}, \mathbf{n}) = \frac{4}{(l * s^2)}\max(\frac{s}{2} - \|(\mathbf{x} - \mathbf{c}) \cdot \mathbf{n}\|, 0).$$

Intuitively, this is a truncated triangular kernel with preferred orientation $\mathbf{n}$. We implement a single roof kernel, a concatenation of two roof kernels with orthogonal orientations as Eq.(3.1.5) and a combination of the same two roof kernels as Eq.(3.1.6) to track a chalk box as shown in Fig. 3.1.1. The surfaces of value $1 - \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}}\|^2$ w.r.t $\Delta\mathbf{c}$ generated by these three methods in one of the tracking iterations are plotted in Fig. 3.1.2.

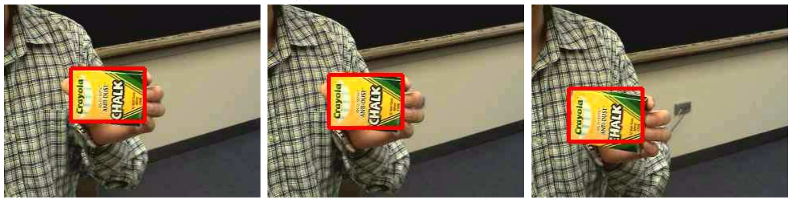
It is now clear that because of the rank deficiency, a single kernel cannot perceive the changes of $\Delta\mathbf{c}$ in some specific directions. While concatenated or combined kernels can well approximate the neighborhood values and ensure to find the $\Delta\mathbf{c}$ minimizing the error $\|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}}\|$, (in Figure 3.1.2, that is maximizing



(a) Single kernel



(b) Kernel concatenation



(c) Kernel combination

Figure 3.1.1: A comparison of single kernel (top row), kernel concatenation (middle row), and kernel combination (bottom row).

$1 - \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}}\|^2$). Concatenated kernels and combined kernels have shown similarly better performance.

However, although these two multiple kernel methods may outperform the single kernel method, neither of them provides a principled way of designing multiple kernels.

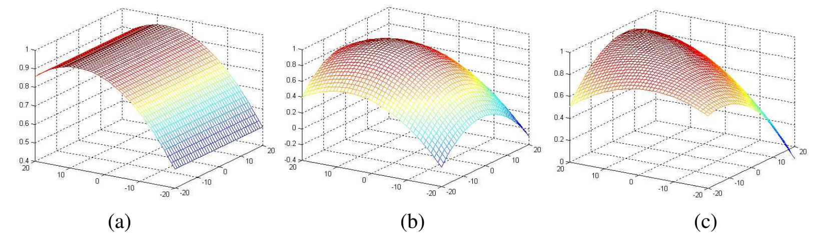


Figure 3.1.2: The surfaces of $1 - \|\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}}\|^2$ of (a) single kernel, (b) kernel concatenation, and (c) kernel combination.

3.2 Multiple Collaborative Kernels

As shown by the examples in Sec. 3.1, it is clear that we expect better performance than single kernel methods by using multiple kernels in the measurement process, Eq.(3.1.2). Based on the kernel observability Theorem, we notice that most existing multiple kernel methods [7, 17], including kernel concatenation and kernel combination, do not utilize the state constraints, Eq.(3.1.3), which should also be used to cope with the rank deficiency. The neglect of the state constraints will largely limit the applicability of these methods, especially for complex objects and motions. This is also one reason that holds the kernel methods back from tracking multiple targets, since simply assigning independent kernels on multiple targets is unlikely to solve the problem.

A new scheme, *multiple collaborative kernels*, is proposed in this section by exploiting the state equation $\Omega(\mathbf{x}) = 0$. We show that this is also an efficient way to improve the "observability" of the tracking system (Sec. 3.2.1), by utilizing appropriate system description (Sec. 3.2.2). Our analysis also reveals the "collaboration" of multiple kernels that makes possible efficient computation (Sec. 3.2.3).

3.2.1 Enhancing the observability

To start with, an obvious and commonly encountered prototype of $\Omega(\mathbf{x}) = 0$ for multiple targets would be the *structural constraint*. Taking a rigid rod as a simple example, see Fig. 3.2.1, we now show the improved "kernel-observability".

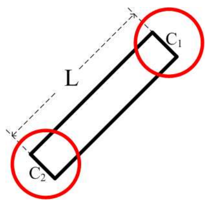


Figure 3.2.1: The length constraint on a rod.

Considering the slim shape of the rod, it is difficult to track it with one simple symmetric kernel. Alternatively, we can relax its motion by representing it as the joint motion of the two ends, while enforcing the length constraint on this relaxed (higher-dimensional) motion. Two simple symmetric kernels take charge of the two ends respectively. The benefit of doing this, besides recovering the rod position, is the estimation of the rod orientation.

By exploring the structural constraint, say, the rod is of fixed length L, we have[2],

$$\|\mathbf{c}_1 - \mathbf{c}_2\|^2 = L^2, \tag{3.2.1}$$

where, $\mathbf{c}_1$ and $\mathbf{c}_2$ are the resulting centers of the kernels placed at the ends. Then the objective function, which jointly considers both of the two kernels, will be formulated as:

$$O(\mathbf{c}_1, \mathbf{c}_2) = \sum_{i=1}^{2} \|\sqrt{\mathbf{q}_i} - \sqrt{\mathbf{p}_i(\mathbf{c}_i)}\|^2 + \gamma \|L^2 - \|\mathbf{c}_1 - \mathbf{c}_2\|^2\|^2,$$

where $\mathbf{q}_1$, $\mathbf{p}_1(\mathbf{c}_1)$ are the target model and the measured candidate associated with one of the ends, similarly with $\mathbf{q}_2$ and $\mathbf{p}_2(\mathbf{c}_2)$. This formulation compromises the *feature similarities* and the *structural constraint*, with γ being the tradeoff. By linearizing it at $(\mathbf{c}_1, \mathbf{c}_2)$, we have a linear system (with state equation and

[2]This simple constraint is for illustrative purpose. More complex constraint will be readily incorporated.

measurement equation):

$$\begin{cases} l &= \mathbf{G}\begin{bmatrix}\Delta\mathbf{c}_1\\ \Delta\mathbf{c}_2\end{bmatrix} \\ \mathbf{y} &= \mathbf{M}\begin{bmatrix}\Delta\mathbf{c}_1\\ \Delta\mathbf{c}_2\end{bmatrix}\end{cases}, \tag{3.2.2}$$

where

$$\overline{\mathbf{q}} = \begin{bmatrix}\mathbf{q}_1\\ \mathbf{q}_2\end{bmatrix}, \overline{\mathbf{p}} = \begin{bmatrix}\mathbf{p}(\mathbf{c}_1)\\ \mathbf{p}(\mathbf{c}_2)\end{bmatrix}, \mathbf{y} = \begin{bmatrix}\sqrt{\mathbf{q}_1} - \sqrt{\mathbf{p}_1(\mathbf{c}_1)}\\ \sqrt{\mathbf{q}_2} - \sqrt{\mathbf{p}_2(\mathbf{c}_2)}\end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix}\mathbf{M}_1 & 0\\ 0 & \mathbf{M}_2\end{bmatrix},$$

$$\mathbf{M}_i = \frac{1}{2}\mathtt{diag}(\mathbf{p}(\mathbf{c}_i))^{-\frac{1}{2}}\mathbf{U}_i^T\mathbf{J}_{\mathbf{K}}(\mathbf{c}_i), \quad i = 1, 2$$

$$\mathbf{G} = 2\begin{bmatrix}(\mathbf{c}_1 - \mathbf{c}_2)^T & (\mathbf{c}_2 - \mathbf{c}_1)^T\end{bmatrix},$$

$$l = L^2 - \|\mathbf{c}_1 - \mathbf{c}_2\|^2.$$

Based on the kernel observability Theorem in Sec. 3.1, the observability of this formulation is given by checking $\mathtt{rank}(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})$. This is equivalent to the column rank of $\begin{bmatrix}\mathbf{M}\\ \sqrt{\gamma}\mathbf{G}\end{bmatrix}$, which will be no less than that of $\mathbf{M}$.

Then, we can generalize the above idea by considering multiple kernels with a certain structural constraint $\Omega(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w) = 0$. The objective function will thus have the form,

$$\begin{aligned} O(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w) &= \sum_{i=1}^{w} \|\sqrt{\mathbf{q}_i} - \sqrt{\mathbf{p}_i(\mathbf{c}_i)}\|^2 \\ &+\gamma\|\Omega(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w)\|^2. \end{aligned} \tag{3.2.3}$$

After the linearization w.r.t. $\Delta\mathbf{c}_1, \Delta\mathbf{c}_2, \ldots, \Delta\mathbf{c}_w$, we have the following general system state equation and measurement equation:

$$\begin{cases} l &= \mathbf{G}\Delta\overline{\mathbf{c}} \\ \mathbf{y} &= \mathbf{M}\Delta\overline{\mathbf{c}} \end{cases}, \tag{3.2.4}$$

where

$$\begin{aligned} &\Delta\overline{\mathbf{c}} = \begin{bmatrix} \Delta\mathbf{c}_1 \\ \Delta\mathbf{c}_2 \\ \cdots \\ \Delta\mathbf{c}_w \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \sqrt{\mathbf{q}_1} - \sqrt{\mathbf{p}(\mathbf{c}_1)} \\ \sqrt{\mathbf{q}_2} - \sqrt{\mathbf{p}(\mathbf{c}_2)} \\ \cdots \\ \sqrt{\mathbf{q}_w} - \sqrt{\mathbf{p}(\mathbf{c}_w)} \end{bmatrix}, \\ &\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & 0 & 0 \\ 0 & \mathbf{M}_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{M}_w \end{bmatrix}, \\ &\mathbf{G} = \begin{bmatrix} \frac{\partial\Omega}{\partial\mathbf{c}_1} & \frac{\partial\Omega}{\partial\mathbf{c}_2} & \cdots & \frac{\partial\Omega}{\partial\mathbf{c}_w} \end{bmatrix}, \\ &l = -\Omega(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w). \end{aligned} \tag{3.2.5}$$

Similarly, the unique motion can be estimated, provided that $\begin{bmatrix} \mathbf{M} \\ \sqrt{\gamma}\mathbf{G} \end{bmatrix}$ has full column rank.

Now, it is worth pointing out that without the introduced constraint $\Omega(\cdot)$, i.e., $\mathbf{G} = \mathbf{0}$, the solution will be reduced to

$$\mathbf{y} = \mathbf{M}\Delta\bar{\mathbf{c}}, \tag{3.2.6}$$

which is equivalent to solving the w kernel tracking problems *independently*, requiring $\mathbf{M}$ to have full column rank, i.e., every kernel needs to be observable.

The advantage of the collaborative kernels is that it does not require all the kernels to be fully observable. Even if some of the kernels get bad, e.g., distracted by the clutters, the other kernels may still be able to "pull" the ill-behaved kernels back to the track according to the inherent constraint embedded in Eq.(3.2.4). As long as $(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})$ is of full rank, our method can tolerate those unobservable kernels. In theory, such a good property is guaranteed by the fact that $\texttt{rank}(\begin{bmatrix} \mathbf{M} \\ \sqrt{\gamma}\mathbf{G} \end{bmatrix}) \geq \texttt{rank}(\mathbf{M})$.

3.2.2 The system description

The system description can have various forms in different applications. Two prototypes of the system description of multiple kernel placements would be the global *subspace* description and the local *component-wise* description.

In the example of tracking a rod in Sec. 3.2.1, the fixed-length model is actually a component-wise description. It is rigid in its form because it is really difficult to learn a comprehensive model of a moving rod. Assuming one end, e_1, of the rod is fixed in the image, the image coordinates of the other end, e_2, can be any point in the circle, with e_1 being the center and L being the radii. However, given short inter-frame interval, the length between the two ends will not vary much. So, the component-wise description can help multiple kernels search for the optimal displacements of instantaneous motions.

A more general form of the multiple kernel placements would be a subspace model. The subspace model are widely employed in computer vision [11, 21] and graphics [27] for representing the data patterns, which is especially useful when a lower dimensional compact representation is needed for a higher dimensional data. The subspace model is learnable. In [3], the sequence of joint angle trajectories of human gaits are modelled into a lower dimensional data stream generated by an “ARMA” model, which is essentially the projection results onto a learned lower dimensional subspace. In [11], a subspace model is learned as the Active Appearance Model for interest points on the face.

In this section, we give the formulation of the system equation using the subspace model. Given a set of training data or learned prior knowledge of multiple interest points on an object, where we wish to place kernels for tracking, the coordinates of these multiple kernels in all the frames, $\overline{\mathbf{c}} \in \mathcal{R}^{2w}$ can be compactly modelled by a lower d-dimensional subspace $\mathcal{S}^d \subset \mathcal{R}^{2w}$. Obviously, the difference between the coordinates, which is the kernel displacement we seek for, resides in the same subspace as well, $\Delta\overline{\mathbf{c}} \in \mathcal{S}^d$.

It is not our intention to convince the reader that the subspace representation offers an optimal modelling for structured object tracking, we just to illustrate that the subspace model is valid in some common data sets and thus offering more flexibility in designing and implementing multiple collaborative kernels for such tracking tasks (such as a part-based tracker with kernels placed on each part), which cannot be handled by single, independent kernels.

We assume that the multiple kernels are placed on the interest points with coordinates $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_w$. For subspace modelling, a common treatment is to eliminate the translation by subtracting the *mean* vector from those coordinates, for $j = 1, \ldots, w$

$$\mathbf{p}^j = \mathbf{p}_j - \mathbf{p}_{mean},$$

where $\mathbf{p}_{mean} = \frac{\sum_{j=1}^{w} \mathbf{p}_j}{w}$.

In the following, the script, f $(f = 1, 2, \ldots)$, will represent the frame number, e.g. $\overline{\overline{\mathbf{c}}}^f$ denotes $\overline{\overline{\mathbf{c}}}$ at frame f, $[]_f$ represents the vector collected at frame f. After obtaining a set of concatenated vectors of kernel coordinates collected from a series of frames,

$$\overline{\overline{\mathbf{c}}}^f = [\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^w]_f^T = [\mathbf{p}_1 - \mathbf{p}_{mean}, \mathbf{p}_2 - \mathbf{p}_{mean}, \ldots, \mathbf{p}_w - \mathbf{p}_{mean}]_f^T \in \mathcal{R}^{2w}, \tag{3.2.7}$$

we can apply PCA to $\overline{\overline{\mathbf{c}}}^f, f = 1, 2, \ldots$ to obtain a lower d-dimensional subspace representation, $\mathcal{S}^d$. For example, the subspace of the in-plane rigid motion will be of dimension 2. An affine motion model will yield an even higher motion subspace. Then, the concatenated kernel *displacement* in each frame, $\Delta\overline{\overline{\mathbf{c}}}^f, f = 1, 2, \ldots$, resides in the same subspace as well, $\Delta\overline{\overline{\mathbf{c}}}^f \in \mathcal{S}^d$. Recall Eq.(3.2.7), it should be clear that

$$\Delta\overline{\overline{\mathbf{c}}}^f = [\Delta\mathbf{c}^1, \Delta\mathbf{c}^2, \ldots, \Delta\mathbf{c}^w]_f^T = \Delta\overline{\mathbf{c}}^f - \Delta\overline{\mathbf{c}}^f_{mean},$$

$$\Delta\overline{\mathbf{c}}^f = [\Delta\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w]_f^T$$

$$\Delta\overline{\mathbf{c}}^f_{mean} = [\Delta\mathbf{c}_{mean}, \Delta\mathbf{c}_{mean}, \ldots, \Delta\mathbf{c}_{mean}]_f^T$$

where $\Delta\mathbf{c}_{mean} = \frac{\sum_{j=1}^{w}\Delta\mathbf{c}_j}{w}$. The notation $\Delta\overline{\overline{\mathbf{c}}}$, which has the mean vector subtracted, is used here to differentiate with $\Delta\overline{\mathbf{c}}$, the vector without mean subtraction, in Sec. 3.2.1.

This brings a practical constraint on $\Delta\overline{\overline{\mathbf{c}}}^f$ when we solve for the kernel displacements in each frame f. The imposed subspace constraint, or the **system equation**, is formulated as follows, (we omit frame number f for brevity.)

$$(\mathbf{I} - \mathbf{V}\mathbf{V}^T)(\Delta\overline{\mathbf{c}} - \Delta\overline{\mathbf{c}}_{mean}) = 0, \tag{3.2.8}$$

where $V = [v_1, v_2, \ldots, v_d]$ is the learned orthonormal basis of the model subspace $\mathcal{S}^d$ by using PCA on the set of vectors $\Delta\overline{\overline{\mathbf{c}}}^f$ with $f = 1, 2, \ldots$ being the frame number. The dimension d is determined by checking the steepest drop in the sorted eigenvalues. Thus $\mathbf{V}\mathbf{V}^T$ is the projection matrix to the subspace $\mathcal{S}^d$, and $(\mathbf{I} - \mathbf{V}\mathbf{V}^T)$ represents the residual after the projection. Eq.(3.2.8) actually represents that the vector $\Delta\overline{\mathbf{c}} - \Delta\overline{\mathbf{c}}_{mean}$ reside in the subspace spanned by $\mathbf{V}$.

Combing Eq.(3.2.8) with Eq.(3.2.6), we have

$$\left\{\begin{array}{rcl} 0 & = & (\mathbf{I}-\mathbf{V}\mathbf{V}^T)(\Delta\bar{\mathbf{c}}-\Delta\bar{\mathbf{c}}_{mean}) \\ \mathbf{y} & = & \mathbf{M}\Delta\bar{\mathbf{c}} \end{array}\right. . \tag{3.2.9}$$

It can also be shown that the constraint will improve the kernel-observability, since the solution to Eq.(3.2.9) is

$$\Delta\bar{\mathbf{c}} = (\mathbf{M}^T\mathbf{M}+\gamma(\mathbf{I}-\mathbf{V}\mathbf{V}^T)^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T))^{-1}(\mathbf{M}^T\mathbf{y}+\gamma(\mathbf{I}-\mathbf{V}\mathbf{V}^T)^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T)\Delta\bar{\mathbf{c}}_{mean}). \tag{3.2.10}$$

It is seen that I). the rank of matrix $\mathbf{M}^T\mathbf{M}+\gamma(\mathbf{I}-\mathbf{V}\mathbf{V}^T)^T(\mathbf{I}-\mathbf{V}\mathbf{V}^T)$ is no less than that of $\mathbf{M}$, thus the overall "kernel-observability" is improved; II). the solution Eq.(3.2.10) not only looks for the location to minimize the color histogram difference, but also shows the consent on the subspace description. The effectiveness of subspace modelling and the encouraging tracking performance will be demonstrated in the experiment section.

The global subspace model and the local component-wise model are just two prototypes of the system description, $\Omega(\mathbf{x})=0$, the above paradigm of design for multiple collaborative kernels can be readily extended to other system descriptions with different physical meanings as well, such as the more complicated motion dynamics or the learned motion priors.

3.2.3 The collaboration

The solution to the linear system in our formulation Eq.(3.2.4 for multiple collaborative kernel tracking is given by:

$$\Delta\overline{\mathbf{c}} = (\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})^{-1}(\mathbf{M}^T\mathbf{y} + \gamma\mathbf{G}^T l). \tag{3.2.11}$$

Specifically, for the component-wise description in Sec. 3.2.1,

$$\begin{aligned} \mathbf{G} &= \begin{bmatrix} \frac{\partial\Omega}{\partial\mathbf{c}_1} & \frac{\partial\Omega}{\partial\mathbf{c}_2} & \cdots & \frac{\partial\Omega}{\partial\mathbf{c}_w} \end{bmatrix}, \\ l &= -\Omega(\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_w). \end{aligned} \tag{3.2.12}$$

For the subspace description in Sec. 3.2.2,

$$\begin{aligned} \mathbf{G} &= \mathbf{I} - \mathbf{V}\mathbf{V}^T \\ l &= (\mathbf{I} - \mathbf{V}\mathbf{V}^T)\Delta\overline{\mathbf{c}}_{mean} \end{aligned} \tag{3.2.13}$$

Due to the relaxation of the system states, the dimension of the matrix $(\mathbf{M}^T\mathbf{M}+\gamma\mathbf{G}^T\mathbf{G})$ can be quite large (the sum of motion parameters of all individual kernels). Thus, it is computationally demanding to calculate its inverse. Considering the special structure of $\mathbf{M}$, we obtain a much more efficient method, which precisely reveals the collaboration among multiple kernels.

By applying matrix inversion lemma[3], we can obtain,

$$\Delta\bar{\mathbf{c}} = (\mathbf{I} - \mathbf{D})(\mathbf{M}^T\mathbf{M})^{-1}(\mathbf{M}^T\mathbf{y} + \gamma\mathbf{G}^T l), \tag{3.2.14}$$

where $\mathbf{D} = \gamma(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{G}^T(\gamma\mathbf{G}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{G}^T + \mathbf{I})^{-1}\mathbf{G}$

Providing that $\mathbf{M}^T\mathbf{M}$ is non-singular, this equation means that we can save the computational cost on $(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})^{-1}$ by computing $(\gamma\mathbf{G}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{G}^T + \mathbf{I})^{-1}$ and $(\mathbf{M}^T\mathbf{M})^{-1}$ instead. Generally, the dimensionality of $(\gamma\mathbf{G}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{G}^T + \mathbf{I})$, which equals the number of constraint, is smaller than the parameters to be estimated, i.e., the dimensionality of $(\mathbf{M}^T\mathbf{M} + \gamma\mathbf{G}^T\mathbf{G})$. Moreover, the calculation of $(\mathbf{M}^T\mathbf{M})^{-1}$ is not difficult since it has a block-diagonal structure form (recalling the structure of $\mathbf{M}$ in Eq.(3.2.5)). All of these count to a potential decrease in the computational cost.

Noticing that the solution to the unconstrained problem (i.e., independent kernels) is given by:

$$\Delta\bar{\mathbf{c}}_u = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{y} = \mathbf{M}^\dagger\mathbf{y}, \tag{3.2.15}$$

where $\mathbf{M}^\dagger$ is the pseudo-inverse of $\mathbf{M}$. This unconstrained solution can be calculated easily with linear cost w.r.t. the number of kernels, since $\mathbf{M}$ is a block

[3] $(\mathbf{A} + \mathbf{BD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{DA}^{-1}\mathbf{B} + \mathbf{I})^{-1}\mathbf{DA}^{-1}$, where **A** is a n by n matrix, **B** is a n by m matrix and **D** is a m by n matrix

diagonal matrix. Any single kernel tracking method can be applied here.

Using the unconstrained solution $\Delta\overline{\mathbf{c}}_u$, we can rewrite the solution to the constrained problem, Eq.(3.2.14), as:

$$\Delta\overline{\mathbf{c}} = (\mathbf{I} - \mathbf{D})\Delta\overline{\mathbf{c}}_u + \mathbf{z}(\mathbf{c}), \tag{3.2.16}$$

where $\mathbf{z}(\mathbf{c}) = \gamma(\mathbf{I} - \mathbf{D})(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{G}^T l$. The mechanism of the collaboration among multiple kernels is pronounced: each individual single-kernel tracker follows its designated target (a small part of the entire target of interest) by its own means, and exchanges "corrections" to other single-kernel tracker. Such a collaboration ends up with an equilibrium where the entire target is tracked and the structural constraints among multiple kernels are satisfied.

The collaboration actually suggests a very efficient recursive method of calculating the constrained solution. We can alternate two steps until convergence: first relax the constraints to solve the unconstrained one by Eq.(3.2.15), and then adjust the unconstrained estimates according to Eq.(3.2.16), with less computational cost.

$$\Delta\overline{\mathbf{c}}^{k+1} \longleftarrow (\mathbf{I} - \mathbf{D}^k)[\mathbf{M}(\Delta\overline{\mathbf{c}}^k)]^\dagger\mathbf{y}^k + \mathbf{z}^k, \tag{3.2.17}$$

which is very similar to the fixed point iteration and converges very fast.

This collaborative solution is useful to multiple target tracking. Because we avoid estimating the motion states from the joint parameter space. Instead, we solve the divided problems in the reduced solution space, then applying regularized terms to meet the certain constraint.

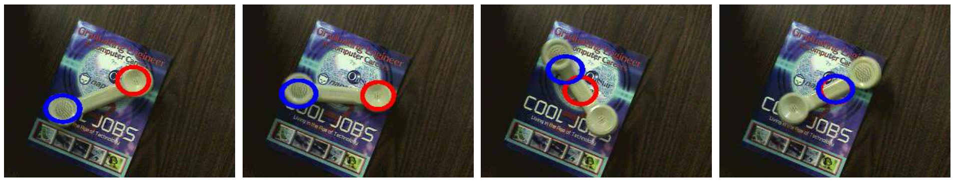
3.3 Experiments

In this section, we report our experiments of the proposed multiple collaborative kernel method to track structured objects and articulated objects, and the comparison to multiple independent kernel tracker.

3.3.1 Tracking structured object

Object with certain spatial structure is a commonplace in many tracking tasks. But some of them, such as a handset or a rod-shaped bottle, cannot be easily handled by the tracker with a single symmetric kernel. See Fig. 3.3.1 and Fig. 3.3.2. Our experiments validate the proposed method of multiple collaborative kernels that can track these targets successfully and to estimate the target orientation as a byproduct.

Fig. 3.3.1 shows 4 sample frames from a sequence of a rotating handset. The histogram in the RGB space is taken as the feature. We first apply two independent


(a) using multiple independent kernels.

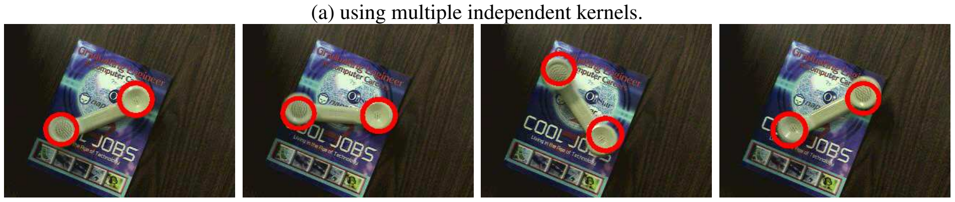
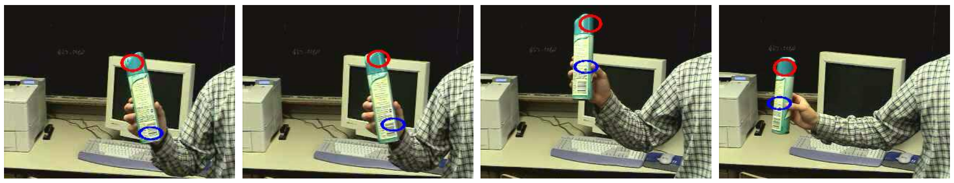

(b) using multiple collaborative kernels.

Figure 3.3.1: Tracking a handset.

normal kernels at both ends of the handset, colored as red and blue, respectively. The result is shown in Fig. 3.3.1(a). We should notice that the motion along the handset is not fully observable for both kernels, and the appearances of the two ends of the handset are identical. The two kernels drift along the handset and eventually lose the track.

With the same kernels but collaborating them based on our method, we introduce a length constraint, $\|\mathbf{c}_1 - \mathbf{c}_2\|^2 = L^2$, with L given by the initialization. The result is shown in Fig. 3.3.1(b). As predicted, the collaboration of the two kernels leads to a successful tracking result. This experiment shows a quite meaningful property of the collaborative kernel approach: although not all the kernels are fully observable, the collaboration can still make the ensemble observable. In all experiments, we set γ in Eq.(3.2.3) to be 1.


(a) using multiple independent kernels.

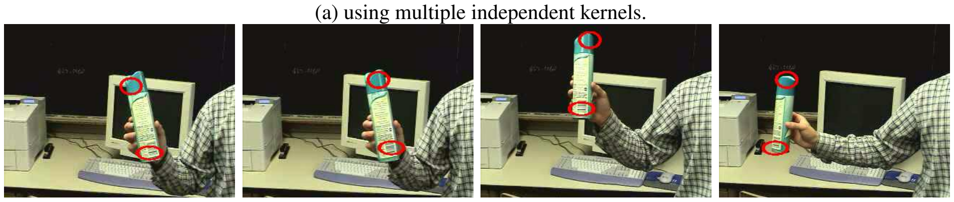
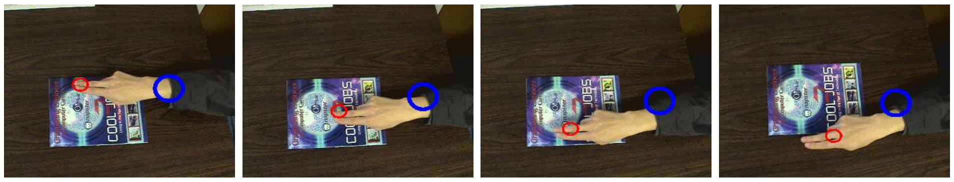

(b) using multiple collaborative kernels.

Figure 3.3.2: Tracking a rod-shaped bottle.

Fig. 3.3.2 shows another experiment on a rod-shaped bottle. We first place two independent normal kernels at the ends. The histogram of H-value of the HSV space is used as the object feature. Sample frames of the result of using independent kernels are shown in Fig. 3.3.2(a). Notice that the motion of the lower-end, indicated by the kernel in blue, is not fully observable, since the image regions in the lower part of the bottle are similar. Thus the blue kernel is vulnerable to distraction when the two kernels function independently. In contrast, the collaboration of the two kernels contributes to a more reliable tracking result, as shown in Fig. 3.3.2(b).

The proposed collaborative scheme also provides another benefit. When a


(a) using multiple independent kernels.

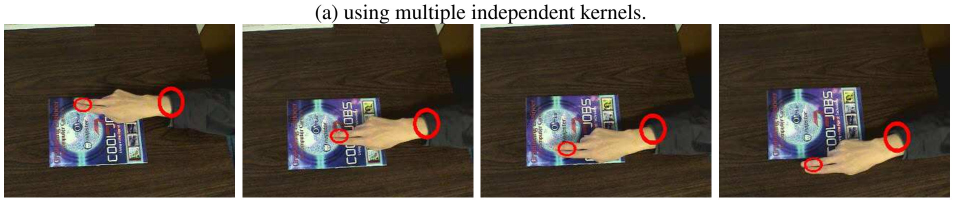

(b) using multiple collaborative kernels.

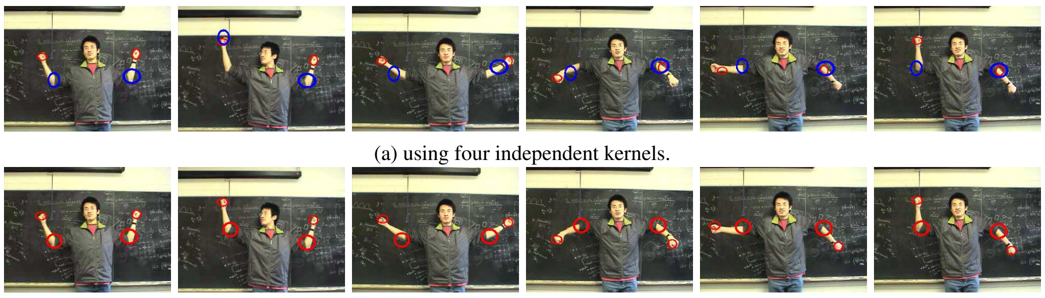
Figure 3.3.3: Tracking a finger.

certain target is our focus-of-attention but unfortunately cannot be stably tracked, we can refer to another easily tracked object as an auxiliary to gain a better result. In Fig. 3.3.3, we aim to track the fingertip in a clutter. By placing a kernel on the easily tracked wrist, we constrain the two kernels with a fixed length. The result of using our method is shown in Fig. 3.3.3(b). In fact, The roles of the object of attention and the auxiliary are interchangeable throughout the process in order to ameliorate the potential tracking failure of either one. Two independent kernels, as shown in Fig. 3.3.3(a), of course are unable to recover from tracking failure in the clutter.

3.3.2 Tracking articulated objects

Another useful application of multiple collaborative kernel tracking is to track articulated targets, such as human body articulation. To the best of our knowledge, this is the first work extending kernel methods into this task.

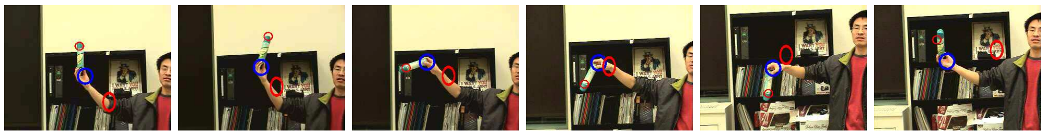
Fig. 3.3.4 shows sample frames of an experiment, in which a person moves his two arms. We apply two pairs of collaborative kernels on the elbows and the hands. The tracking result of our approach is shown in Fig. 3.3.4(b). On the contrary, the method based on four independent kernels leads to a much inferior performance, as shown in Fig. 3.3.4(a).


(a) using four independent kernels.

(b) using two pairs of collaborative kernels.

Figure 3.3.4: Tracking the articulated body with two arms.

Another experiment on an articulated structure consisting of an arm and a bottle in hand is shown in Fig. 3.3.5. We apply three kernels to the elbow, the hand and one end of the bottle, respectively. Compared with the result yielded by independent kernels (in Fig. 3.3.5(a)), two pairs of collaborative kernels (elbow & hand, hand & bottle tip) provide a much more robust performance as shown in Fig. 3.3.5(b).


(a) using three independent kernels.

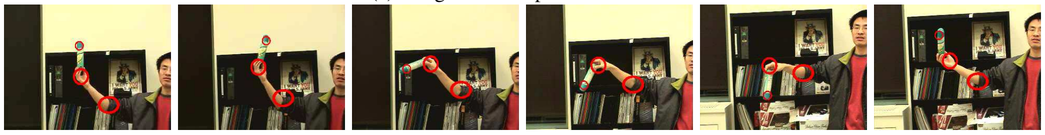

(b) using two pairs of collaborative kernels.

Figure 3.3.5: Tracking an articulated structure.

The structural constraint used here serves as a basic means facilitating the implementation of multiple collaborative kernels on more complex tracking tasks.

3.3.3 Using the subspace model

In this section, we demonstrate the advantages provided by using the subspace constraint model.

First, we need to learn the subspace model from the training data. Fig. 3.3.6 shows the sample image, in which a static box is being viewed by a moving camera. We manually labelled three interest points denoted by red "x" through a sequence of 50 frames. A subspace model is then obtained by applying PCA on the concatenated vectors of kernel positions $\overline{\overline{\mathbf{c}}} = \overline{\mathbf{c}} - \overline{\mathbf{c}}_{mean}$ (with mean vector subtracted). The dimension of the subspace is determined by examining the sharpest drop in the sorted eigenvalues.

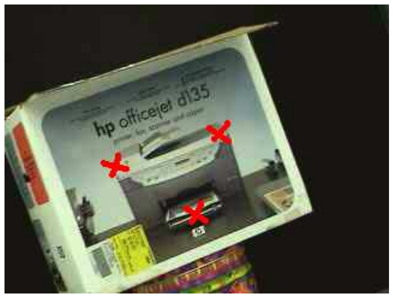
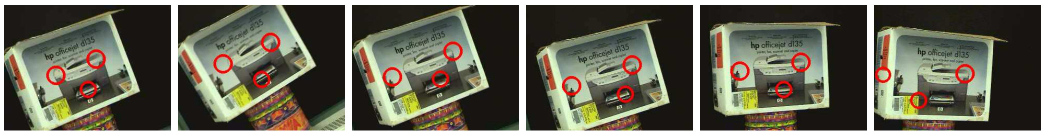


Figure 3.3.6: Labelled interest points for subspace model learning.

Then, the solution Eq.(3.2.16) is used to guide the collaborative kernels for tracking. This formula makes clear a two-step approach. One is an unconstrained kernel shifting: $\Delta\overline{\mathbf{c}}_u$, and the next is a constrained subspace regularization. In experiment, we implement this two-step method for computational efficiency, as was described in Sec. 3.2.3. If the resulting kernel displacement deviates a lot from the subspace model, a mapping to the learned subspace is taken for regularization purpose.

For all of the sequences, we also test the single independent kernel method. Both methods are applied for every other frame. The comparison result for this "box" sequence is shown in Fig. 3.3.7. We can see that in presence of fast scene changes, single kernels are more vulnerable to distractions and are more prone to lose the track. In contrast, the enforced subspace constraint is capable to stabilize the kernel movements, thus having an overall better ability to avoid distractions for each kernel.


(a) using three independent kernels.

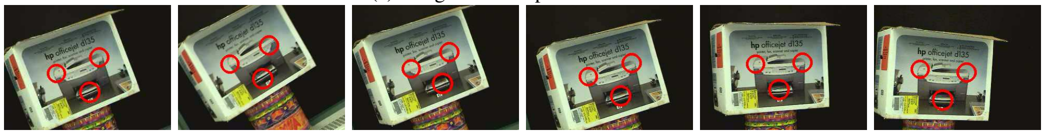
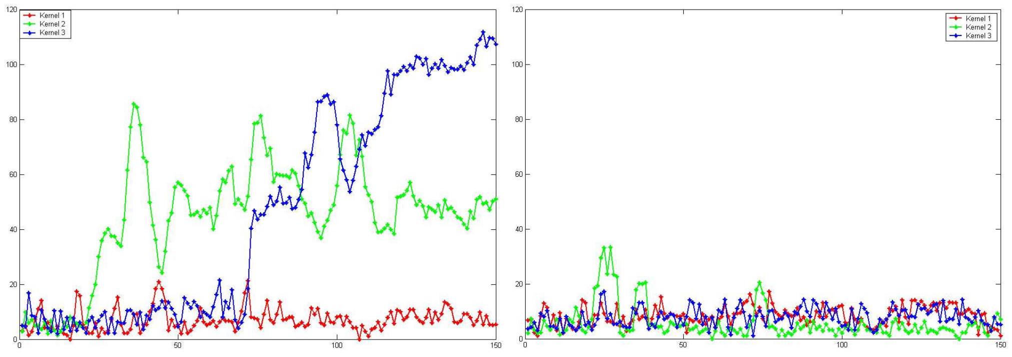

(b) using three collaborative kernels under subspace constraint.

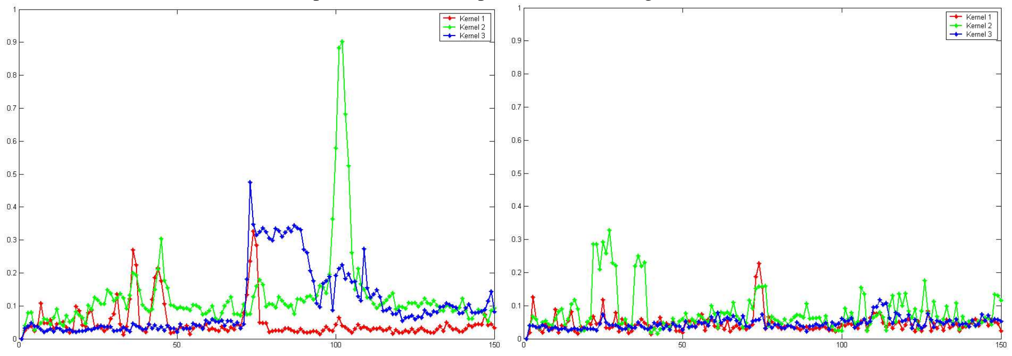
Figure 3.3.7: Tracking three parts of interest on a box.

We have also compared the results of single and collaborative kernels against the ground truth, which is obtained by manually labelling. Fig. 3.3.8(a) shows the error of kernel positions obtained through 150 frames. Fig. 3.3.8(b) shows the error of histogram matching in the same sequence. The robustness of the collaborative kernels is shown.

Fig. 3.3.9 shows the result of another moving box sequence. The subspace model is learned from a sequence of 50 frames. It can be seen that the single kernels are subject to drifting along a certain direction. For instance, look at the



(a) Errors in kernel position. Left: independent kernels. Right: collaborative kernels.

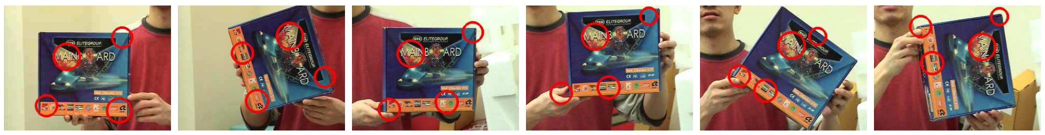


(b) Errors in histogram matching. Left: independent kernels. Right: collaborative kernels.

Figure 3.3.8: Error comparison of single and collaborative kernels against the ground truth.

orange stripe at the bottom of the box. The kernels cannot observe the precise movement along this stripe because all the locations in this stripe yield the similar histogram as that of the original target. For comparison, the subspace constraint is shown to be able to discriminate and regularize the invalid kernel positions, thus more robust result is obtained.

Fig. 3.3.10 shows the result for tracking the head and shoulders of a person.


(a) using four independent kernels.

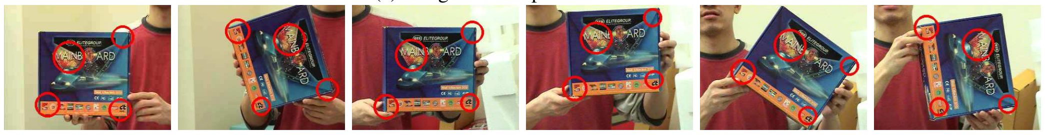
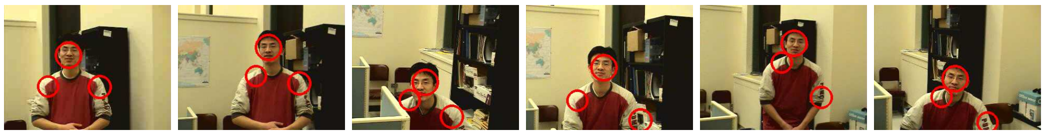

(b) using four collaborative kernels under subspace constraint.

Figure 3.3.9: Tracking four parts of interest on a box.

As shown in Fig. 3.3.10(a), the performance of independent kernels is not warranted by neglecting the spatial constraint among them. Drift and deviation are observed. In contrast, since the training data we collected for subspace modelling admits noise measurement and variations, to some extent, in the scale, the subspace model can help to tolerate some scale changes of tracked object, e.g., the person went down or came closer to the camera, as shown in Fig. 3.3.10(b). A more stable performance is obtained.


(a) using three independent kernels.

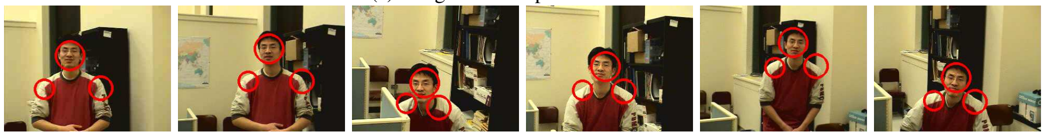
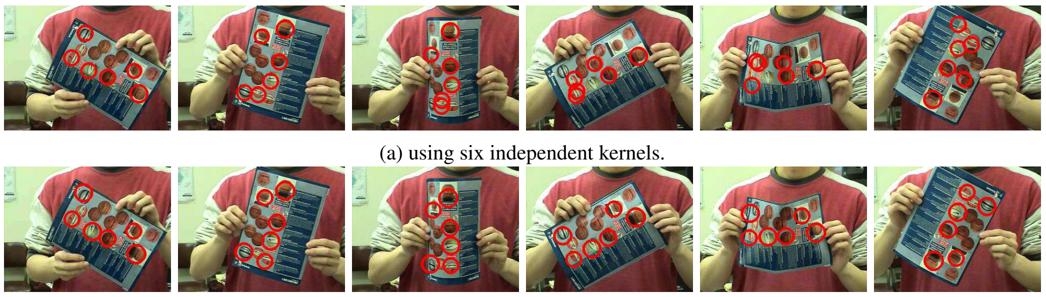

(b) using three collaborative kernels under subspace constraint.

Figure 3.3.10: Tracking the head and shoulders.

A more challenging task is shown in Fig. 3.3.11. We want to track some interest regions on a magazine cover, which is undergoing deformation. A subspace model is learned from 75 frames. Fig. 3.3.11(a) shows the tracking result of six independent kernels. Their ability of tolerating distortion is poor. Several kernels easily drift away. The collaborative kernel tracking, as shown in Fig. 3.3.11(b), achieves more robust performance. By framing in the subspace model, both histograms matching and subspace model regularizing contribute to stabilize the kernel positions against large distortions. The resulting kernels can well grasp this structured magazine cover.



(a) using six independent kernels.

(b) using six collaborative kernels under subspace constraint.

Figure 3.3.11: Tracking a magazine cover with deformation.

To summarize, by using multiple kernels, we can ease the burden of representing and tracking a holistic object, possibly with some distortions or deformations, by using a set of collaborative kernels. This is a feasible approach [1, 18], although some more challenging problems, such as model updating, are worth being further studied.

3.4 Remarks

In this chapter, a criterion is obtained on the issue of “kernel-observability”, which leads to a principled way of kernel design with prevention of singularity in kernel based tracking problems. Based on this, a multiple collaborative kernel tracking scheme is proposed. Different from the most existing kernel based algorithms, which are confined by independent kernels and single target, we show that by exploiting the inherent relationship among multiple kernels, not only the “kernel-observability” is improved, but also the applicability of the kernel based methods is naturally extended to cope with articulated targets and complex motions. This helps to gain more insight into the role that kernel plays in the tracking problems.

However, the geometric constraint used in this chaper is rigid in its current form. The subspace constraint offers more modelling power and tolerance, but still, lacks an update scheme, which alone is an interesting research topic in visual tracking. The focus of our future work will be exploring how to incorporate richer system models to account for more complicated motions and how to make the kernel design adaptable to various environmental changes.

Chapter 4

Efficient Optimal Kernel Placement for Reliable Visual Tracking

4.1 Optimal Single Kernel Placement

As mentioned in Sec. 2.2.2, the ill-conditioned case of unstableness in tracking may notably deteriorate the tracking performance. Different kernel placement can yield quite different stablilities in tracking. In this section, we give more detailed analysis into such a case and propose a criterion to select optimal locations to place kernels, which avoids the ill-conditioning to the largest extent. The analysis is more easily approachable under the formulation of closed-form tracking in

Sec. 2.1.3. For brevity, notations, such as the histogram representation and the solution of estimated motion for tracking, are referred to Sec. 2.1.3. The analytical result for optimal kernel placement is equivalently applicable to that of mean shift tracking in Sec. 2.1.2.

4.1.1 Applying the condition theory

To analyze the stableness, or say, the sensitivity, of the solution $\Delta\mathbf{c}$ for kernel placement, we first refer to the condition theory.

To solve **x** from a linear equation,

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

besides requiring **A** to be invertible, it is also expected that the solution is numerically stable. The analysis of how sensitive the **x** is, given changes in **b**, can be achieved by examining the condition number defined as,

$$\kappa(\mathbf{A}) = \|A\|\|A^{-1}\|.$$

For example, when $2-$norm is used, $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \sigma_1(\mathbf{A})/\sigma_n(\mathbf{A})$, which is the ratio between the largest and the smallest singular value.

For a single kernel, we need to calculate the motion parameter $\Delta\mathbf{c}$ from $\mathbf{M}\Delta\mathbf{c} = \sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$. The solution is

$$\Delta\mathbf{c} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T(\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}).$$

So, $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ should be considered as a whole entity, which tells how sensitive the $\Delta\mathbf{c}$ is, given small changes in $\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$.

Since $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ is not a square matrix, its "condition number" is not well defined. However, considering the essence of this problem, if we take SVD of the $2 \times m$ matrix $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ as $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T = \mathbf{U}\Sigma\mathbf{V}^T$.

We would expect that the 2 singular values in Σ be comparable to each other, such that the $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ is equally sensible in both directions of its two orthonormal singular vectors. Otherwise, if the two singular values are unbalanced, a fluctuation in $\sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$ caused by noise will change the solution $\Delta\mathbf{c}$ significantly along the singular vector corresponding to the larger singular value, and negligibly along the singular vector corresponding to the smaller singular value, bringing in undesirable numerical instability, and such a region is generally considered to be a bad placement of the kernel.

Notice that

$$(\mathbf{M}^T\mathbf{M})^{-1} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T((\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T)^T = \mathbf{U}\Sigma^2\mathbf{U}^T,$$

and assume σ_1 and σ_2 are two singular values of $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$, it is easy to verify that

$$\kappa_2((\mathbf{M}^T\mathbf{M})^{-1}) = (\sigma_1/\sigma_2)^2.$$

We also have $\kappa_2(\mathbf{M}^T\mathbf{M}) = \kappa_2((\mathbf{M}^T\mathbf{M})^{-1})$. In view of this, the sensitivity evaluation of $(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$ is just equivalent to inspecting the condition number of $(\mathbf{M}^T\mathbf{M})$, since $\kappa_2(\mathbf{M}^T\mathbf{M})$ monotonically increases/decreases when σ_1/σ_2 increases/decreases.

So, the *criterion* for a reliable kernel tracking is: we need to put the kernel to such a place that the condition number of $\mathbf{M}^T\mathbf{M}$ is minimized.

$$\min_{\mathbf{c}} \kappa_2(\mathbf{M}^T\mathbf{M}). \tag{4.1.1}$$

4.1.2 Interpretation of the condition number criterion using the 2-norm

Here, we give an intuitive interpretation of the condition number criterion using the 2-norm, which requires to evaluate the singular values of $\mathbf{M}^T\mathbf{M}$. Actually, evaluating the singular values of $\mathbf{M}$ doesn't affect the analytical result.

In the following, $\mathbf{x}_i$ represents a data point with index i, while $\mathbf{x}_i^j$ denotes point i of color j. For the problem of n points within the kernel range and m color bins. By recalling Eq.(2.1.19),

$$\mathbf{M} = \frac{1}{2}\mathtt{diag}(\mathbf{p}(\mathbf{c}))^{-\frac{1}{2}}\mathbf{U}^T\mathbf{J}_{\mathbf{K}}(\mathbf{c}).$$

The i^{th} row of the $n \times 2$ matrix $\mathbf{J}_{\mathbf{K}}$ is $(\mathbf{x}_i - \mathbf{c})g\left(\|\frac{\mathbf{x}_i-\mathbf{c}}{h}\|^2\right)$, with $g(\cdot) = -k'(\cdot)$ and $k'(\cdot)$ being the profile of the kernel $\mathbf{K}$. Then, by left multiplying the $m \times n$ sifting matrix $\mathbf{U}^T$, the resulting $m \times 2$ matrix, denoted as $\mathbf{D} = \mathbf{U}^T\mathbf{J}_{\mathbf{K}}(\mathbf{c})$, has the meaning that the j^{th} row of $\mathbf{D}$ is the sum of $\mathbf{x}_i^j - \mathbf{c}$ weighted by $g\left(\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\|^2\right)$ for all pixels $\mathbf{x}_i$ of color j, $i = 1, \ldots, n, j = 1, \ldots, m$.

As for $\mathbf{M} = \frac{1}{2}\mathtt{diag}(\mathbf{p}(\mathbf{c}))^{-\frac{1}{2}}\mathbf{D}$, we can see that each row of $\mathbf{M}$ is just the normalization of the corresponding row in $\mathbf{D}$ by a factor of $2\mathbf{p}(\mathbf{c})^{\frac{1}{2}}$, thus giving a particular constraint on $\Delta\mathbf{c}$,

$$\left[\frac{1}{2\sqrt{\mathbf{p}_j}}\sum_i(\mathbf{x}_i^j-\mathbf{c})g\left(\left\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\right\|^2\right)\right]\Delta\mathbf{c}=\sqrt{\mathbf{q}_j}-\sqrt{\mathbf{p}_j}. \tag{4.1.2}$$

The intuition of the LHS of this equation, i.e., the j^{th} row of **M**, is that we sum all the displacement vector $\mathbf{x}_i^j - \mathbf{c}$ of color j, which are weighted by $g\left(\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\|^2\right)$, and then the summation is scaled by $\frac{1}{2\sqrt{\mathbf{p}_j}}$. Denote this result as $[d_x^j \quad d_y^j]$, which can be effectively considered as the *center of mass* of all pixels of color j.

Since a good kernel placement is featured by a **M** with comparable singular values, this requires that all the rows of **M**, $[d_x^j \quad d_y^j], j = 1\ldots, m$ well span the 2D space. The corresponding situation is that all the center of masses of the color components should be distributed evenly around the center **c**.

4.1.3 An equivalent condition number

In practice, 2-norm condition number is not straightforward to compute. In this section, we introduce another form of condition number, being equivalent to the 2-norm condition number when the matrix is 2×2 symmetric positive definite. The new condition number offers a great ease of computation and facilitates an efficient searching algorithm for optimal kernel placement, as will be derived as follows.

The Schatten 1-norm [2][19][26] is defined as,

$$\|\mathbf{A}\|_S = \sum \sigma_i,$$

where $\sigma_1, \ldots, \sigma_n$ are the singular values of $\mathbf{A}$. When $\mathbf{A}$ is a symmetric positive definite matrix, we can have,

$$\begin{aligned} \|\mathbf{A}\|_S &= \sum \sigma_i = \text{trace}(\mathbf{A}), \\ &\prod \sigma_i = \det(\mathbf{A}). \end{aligned} \tag{4.1.3}$$

Given a 2×2 symmetric positive definite matrix $\mathbf{A}$=$\{a_{ij}\}$, we can then have a *closed form* expression of S-norm condition number as,

$$\begin{aligned} \kappa_S(\mathbf{A}) &= \|\mathbf{A}\|_S \|\mathbf{A}^{-1}\|_S = \text{trace}(\mathbf{A})\text{trace}(\mathbf{A}^{-1}) \\ &= \text{trace}(\mathbf{A})\frac{\text{trace}(\mathbf{A})}{\det(\mathbf{A})} = \frac{(a_{11}+a_{22})^2}{a_{11}a_{22}-a_{12}a_{21}}. \end{aligned} \tag{4.1.4}$$

And equivalently,

$$\kappa_S(\mathbf{A}) = \frac{(\sigma_1 + \sigma_2)^2}{\sigma_1\sigma_2}. \tag{4.1.5}$$

According to [16], any two condition numbers $\kappa_\alpha(\mathbf{A})$ and $\kappa_\beta(\mathbf{A})$ are equivalent in that constants c_1 and c_2 can be found for which

$$c_1\kappa_\alpha(\mathbf{A}) \le \kappa_\beta(\mathbf{A}) \le c_2\kappa_\alpha(\mathbf{A}).$$

For example, $\frac{1}{n}\kappa_2(\mathbf{A}) \leq \kappa_1(\mathbf{A}) \leq n\kappa_2(\mathbf{A})$, for $\mathbf{A}\in \mathcal{R}^{n\times n}$. Here, we can show that,

Proposition 1 $\kappa_2(\mathbf{A}) \leq \kappa_S(\mathbf{A}) \leq n^2\kappa_2(\mathbf{A})$, *if matrix* $\mathbf{A}$ *is* $n \times n$ *symmetric positive definite.*

Proof:

Denote $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ are n sorted singular values of $\mathbf{A}$.

$$\kappa_S(\mathbf{A}) = \|\mathbf{A}\|_S \|\mathbf{A}^{-1}\|_S$$

$$\|\mathbf{A}\|_S = \text{trace}(\mathbf{A}) = \sum_{i=1}^{n} \sigma_i, \quad \|\mathbf{A}^{-1}\|_S = \text{trace}(\mathbf{A}^{-\mathbf{1}}) = \sum_{i=1}^{n} \frac{1}{\sigma_i}$$

Since,

$$\sum_{i=1}^{n} \sigma_i \geq \sigma_1, \qquad \sum_{i=1}^{n} \frac{1}{\sigma_i} \geq \frac{1}{\sigma_n},$$

so,

$$\kappa_S(\mathbf{A}) \geq \frac{\sigma_1}{\sigma_n} = \kappa_2(\mathbf{A})$$

Also,

$$\sum_{i=1}^{n} \sigma_i \leq n\sigma_1, \qquad \sum_{i=1}^{n} \frac{1}{\sigma_i} \leq \frac{n}{\sigma_n},$$

we have,

$$\kappa_S(\mathbf{A}) \leq n\sigma_1 \cdot \frac{n}{\sigma_n} = n^2\kappa_2(\mathbf{A})$$

Therefore, $\kappa_2(\mathbf{A}) \leq \kappa_S(\mathbf{A}) \leq n^2\kappa_2(\mathbf{A})$.■

Besides this interlacing property, which is existed for all the condition numbers, we can further show that,

Proposition 2 *κ_S is monotonically* increasing/decreaing *when κ_2* increases/decreases*, if the matrix is 2 $\times$ 2 symmetric positive definite.*

Proof:

κ_S is monotonic with κ_2 as long as the derivative of κ_S w.r.t κ_2 does not change sign. We know that $\kappa_2 \geq 1$ and according to Eq.(4.1.5),

$$\begin{aligned} \kappa_S &= \tfrac{(\sigma_1+\sigma_2)^2}{\sigma_1\sigma_2} = \tfrac{\sigma_1^2+2\sigma_1\sigma_2+\sigma_2^2}{\sigma_1\sigma_2} \\ &= \tfrac{\sigma_1}{\sigma_2} + 2 + \tfrac{\sigma_2}{\sigma_1} = \kappa_2 + 2 + \tfrac{1}{\kappa_2} \end{aligned}$$

so,

$$\frac{d\kappa_S}{d\kappa_2} = 1 - \frac{1}{\kappa_2^2} > 0 \quad \forall \kappa_2 \geq 1■$$

Therefore, since $\mathbf{M}^T\mathbf{M}$ is 2 $\times$ 2 symmetric positive definite, if we find the local/global minimum of its κ_S, we in fact find the local/global minimum of its κ_2. This nice property ensures that the good kernel measured by κ_S is just the good kernel measured by κ_2. Because computing κ_S only involves element-wise calculation, it is much more convenient and efficient than κ_2, which is non-analytical.

So the claim is that, κ_S is an equivalent substitute for κ_2 in all the cases [5][24][25][28] when the 2-norm condition number of a 2×2 covariance matrix needed to be evaluated.

To summarize, given

$$\mathbf{M} = \begin{bmatrix} d_x^1 & d_y^1 \\ \vdots & \vdots \\ d_x^m & d_y^m \end{bmatrix},$$

where

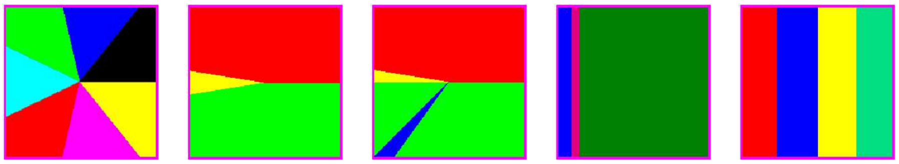
$$[d_x^j \quad d_y^j] = \left[\frac{1}{2\sqrt{\mathbf{p}_j}} \sum_{i,b(\mathbf{x}_i)=j} (\mathbf{x}_i^j - \mathbf{c}) g\left(\left\| \frac{\mathbf{x}_i^j - \mathbf{c}}{h} \right\|^2 \right) \right], \qquad (4.1.6)$$

is the weighted sum of the displacement vectors of all pixels of color j, or called the center of mass of color component j.

We can compute in closed-form

$$\begin{aligned} \kappa_S(\mathbf{M}^T\mathbf{M}) &= \|(\mathbf{M}^T\mathbf{M})\|_S \|(\mathbf{M}^T\mathbf{M})^{-1}\|_S \\ &= \frac{(\sum (d_x^j)^2 + \sum (d_y^j)^2)^2}{\sum (d_x^j)^2 \sum (d_y^j)^2 - (\sum (d_x^j d_y^j))^2}. \end{aligned} \qquad (4.1.7)$$

The region with a small $\kappa_S(\mathbf{M}^T\mathbf{M})$ is the good choice for kernel placement, on which the stability of tracking will be better than those regions with larger condition numbers. Fig. 4.1.1 shows some synthesized image patterns and their κ_2, κ_S. The first one has the lowest condition number, coinciding the interpretation given in Sec. 4.1.2. Fig. 4.1.2 shows the κ_2, κ_S and their differences evaluated in two image regions. It is observed that they exhibit the same pattern of ridges and valleys, and their differences are small compared with their own magnitudes.



$\kappa_2 = 1.0,$ $\kappa_S = 4.0,$ $\kappa_2 = 8.3,$ $\kappa_S = 10.4,$ $\kappa_2 = 11.1,$ $\kappa_S = 13.2,$ $\kappa_2 = 10301,$ $\kappa_S = 10303,$ $\kappa_2 = \infty$ $\kappa_S = \infty$

Figure 4.1.1: Synthesized image patterns and their κ_2, κ_S.

4.1.4 Find optimal kernel placement efficiently

In practice, only obtaining the criterion for optimal kernel placement is insufficient, since it is not attractive to exhaustively evaluate this criterion all over the image. In this section, we derive a gradient descent algorithm, which can efficiently find good placement for kernels.

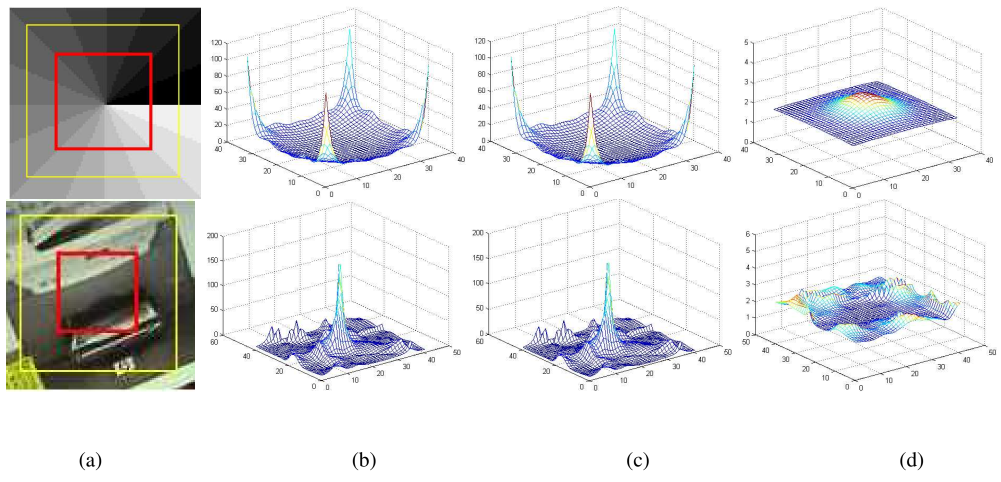


Figure 4.1.2: column (a): yellow: image region, red: kernel. column (b), (c), and (d) are κ_2, κ_S and $\kappa_S - \kappa_2$ evaluated in the region of column (a), respectively.

Notice that in Eq.(4.1.6) and Eq.(4.1.7), the condition number κ_S only involves d_x^j and d_y^j, which are explicitly presented as a function of the kernel position **c**. So, we can compute the *derivative* of $\kappa_S(\mathbf{M}^T\mathbf{M})$ w.r.t the kernel position, **c**. Denote

$$\kappa_S(\mathbf{M}^T\mathbf{M}) = \frac{A^2}{B} = \frac{A^2}{DE - F^2},$$

where

$$A = \textstyle\sum_j \|[d_x^j \quad d_y^j]\|^2 = \sum_j (d_x^j)^2 + \sum_j (d_y^j)^2,$$

$$D = \textstyle\sum_j (d_x^j)^2, \quad E = \sum_j (d_y^j)^2, \quad F = \sum_j (d_x^j d_y^j),$$

$$B = \textstyle\sum_j (d_x^j)^2 \sum_j (d_y^j)^2 - \left(\sum_j (d_x^j d_y^j)\right)^2 = DE - F^2.$$

Here, the goal is to compute

$$\frac{\partial \frac{A^2}{B}}{\partial \mathbf{c}} = \frac{2AB\frac{\partial A}{\partial \mathbf{c}} - A^2\frac{\partial B}{\partial \mathbf{c}}}{B^2} = \frac{2AB\frac{\partial A}{\partial \mathbf{c}} - A^2(D\frac{\partial E}{\partial \mathbf{c}} + E\frac{\partial D}{\partial \mathbf{c}} - 2F\frac{\partial F}{\partial \mathbf{c}})}{B^2}.$$

So, we need $\frac{\partial A}{\partial \mathbf{c}}$, $\frac{\partial D}{\partial \mathbf{c}}$, $\frac{\partial E}{\partial \mathbf{c}}$, and $\frac{\partial F}{\partial \mathbf{c}}$. In practice, we will express $\frac{\partial D}{\partial \mathbf{c}} = [\frac{\partial D}{\partial c_x} \quad \frac{\partial D}{\partial c_y}]$, $\frac{\partial E}{\partial \mathbf{c}} = [\frac{\partial E}{\partial c_x} \quad \frac{\partial E}{\partial c_y}]$, and $\frac{\partial F}{\partial \mathbf{c}} = [\frac{\partial F}{\partial c_x} \quad \frac{\partial F}{\partial c_y}]$. Details can be derived as follows,

$$\begin{aligned}
&\frac{\partial A}{\partial \mathbf{c}} = \sum_j 2[d_x^j \quad d_y^j]\frac{\partial [d_x^j \quad d_y^j]}{\partial \mathbf{c}},\\
&\frac{\partial D}{\partial c_x} = \sum_j 2d_x^j\frac{\partial d_x^j}{\partial c_x}, \qquad \frac{\partial D}{\partial c_y} = \sum_j 2d_x^j\frac{\partial d_x^j}{\partial c_y},\\
&\frac{\partial E}{\partial c_x} = \sum_j 2d_y^j\frac{\partial d_y^j}{\partial c_x}, \qquad \frac{\partial E}{\partial c_y} = \sum_j 2d_y^j\frac{\partial d_y^j}{\partial c_y},\\
&\frac{\partial F}{\partial c_x} = \sum_j \left[\frac{\partial d_x^j}{\partial c_x}d_y^j + d_x^j\frac{\partial d_y^j}{\partial c_x}\right], \qquad \frac{\partial F}{\partial c_y} = \sum_j \left[\frac{\partial d_x^j}{\partial c_y}d_y^j + d_x^j\frac{\partial d_y^j}{\partial c_y}\right].
\end{aligned}$$

where

$$\begin{aligned}
&\frac{\partial [d_x^j \quad d_y^j]}{\partial \mathbf{c}} = \frac{1}{2\sqrt{\mathbf{P}}_j}\left[\sum_i (-1)g\left(\left\|\frac{\mathbf{x}_i^j - \mathbf{c}}{h}\right\|^2\right) + \sum_i (\mathbf{x}_i^j - \mathbf{c})^T\frac{\partial g}{\partial \mathbf{c}}\right],\\
&\frac{\partial d_x^j}{\partial c_x} = \frac{1}{2\sqrt{\mathbf{P}}_j}\left[\sum_i (-1)g\left(\left\|\frac{\mathbf{x}_i^j - \mathbf{c}}{h}\right\|^2\right) + \sum_i (\mathbf{x}_{ix}^j - c_x)\frac{\partial g}{\partial c_x}\right],\\
&\frac{\partial d_x^j}{\partial c_y} = \frac{1}{2\sqrt{\mathbf{P}}_j}\sum_i (\mathbf{x}_{ix}^j - c_x)\frac{\partial g}{\partial c_y},\\
&\frac{\partial d_y^j}{\partial c_x} = \frac{1}{2\sqrt{\mathbf{P}}_j}\sum_i (\mathbf{x}_{iy}^j - c_y)\frac{\partial g}{\partial c_x},\\
&\frac{\partial d_y^j}{\partial c_y} = \frac{1}{2\sqrt{\mathbf{P}}_j}\left[\sum_i (-1)g\left(\left\|\frac{\mathbf{x}_i^j - \mathbf{c}}{h}\right\|^2\right) + \sum_i (\mathbf{x}_{iy}^j - c_y)\frac{\partial g}{\partial c_y}\right].
\end{aligned}$$

When $g()$ is Gaussian kernel, i.e., $g\left(\left\|\frac{\mathbf{x}_i^j - \mathbf{c}}{h}\right\|^2\right) = \exp\left(-\left\|\frac{\mathbf{x}_i^j - \mathbf{c}}{h}\right\|^2\right)$, we have

$$\frac{\partial g}{\partial \mathbf{c}} = \exp\left(-\left\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\right\|^2\right)\frac{2(\mathbf{x}_i^j-\mathbf{c})}{h^2}$$
$$\frac{\partial g}{\partial c_x} = \exp\left(-\left\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\right\|^2\right)\frac{2(\mathbf{x}_{ix}^j-c_x)}{h^2}$$
$$\frac{\partial g}{\partial c_y} = \exp\left(-\left\|\frac{\mathbf{x}_i^j-\mathbf{c}}{h}\right\|^2\right)\frac{2(\mathbf{x}_{iy}^j-c_y)}{h^2}$$

When Epanechnikov kernel is used, i.e., $g(\mathbf{c}) = 1$ and $\frac{\partial g}{\partial \mathbf{c}} = 0$, the above equations will be reduced to much simpler forms as shown below.

$$A = \sum_j \|[d_x^j \;\; d_y^j]\|^2 = \sum_j \frac{1}{4\mathbf{p}_j}\left\|\sum_i(\mathbf{x}_i^j - \mathbf{c})\right\|^2$$
$$D = \sum_j \frac{1}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{ix}^j - c_x)\right]^2$$
$$E = \sum_j \frac{1}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{iy}^j - c_y)\right]^2$$
$$F = \sum_j \frac{1}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{ix}^j - c_x)\sum_i(\mathbf{x}_{iy}^j - c_y)\right]$$

$$\frac{\partial A}{\partial \mathbf{c}} = \sum_j \frac{2}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_i^j - \mathbf{c})\right]\left[\sum_i(-1)\right]$$
$$\frac{\partial D}{\partial c_x} = \sum_j \frac{2}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{ix}^j - c_x)\right]\left[\sum_i(-1)\right], \qquad \frac{\partial D}{\partial c_y} = 0$$
$$\frac{\partial E}{\partial c_x} = 0, \qquad \frac{\partial E}{\partial c_y} = \sum_j \frac{2}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{iy}^j - c_y)\right]\left[\sum_i(-1)\right]$$
$$\frac{\partial F}{\partial c_x} = \sum_j \frac{1}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{iy}^j - c_y)\right]\left[\sum_i(-1)\right], \frac{\partial F}{\partial c_y} = \sum_j \frac{1}{4\mathbf{p}_j}\left[\sum_i(\mathbf{x}_{ix}^j - c_x)\right]\left[\sum_i(-1)\right]$$

Notice that all the above values can be obtained by scanning the pixels in the kernel region only *once*, so the calculation is *easy and efficient*.

Fig. 4.1.3 shows some illustrative examples. In all 3 images in (a), we start from the kernel with red rectangle and end up with the kernel with the green one, led by the gradient-based search. Fig. 4.1.3(b) shows the descending κ_S. Fig. 4.1.3(c) shows the regions covered by the kernel, which is moving along the direction of the gradient towards the good placement.

4.2 Multiple Kernel Placement

As demonstrated in Chapter 3, the kernel based tracking is no longer confined to single kernels. Multiple kernels have several advantages over single kernel. For example, multiple kernels can alleviate the singularity and improve the kernel's observability to the motions [13][17]. Multiple kernels are better at handling tracking an object with complex structure, while a holistic representation based on a single kernel is cumbersome. In such a case, distributing the tracking task into several correlated sub-tasks would be viable. Another benefit is the save of the computation since each sub-task only needs a relatively small kernel.

We think good strategies to place multiple kernels are I) each kernel has a reliable tracking performance, i.e., at a good location, and based on which, II) the structure of the multiple kernels should remain stable through the sequence and

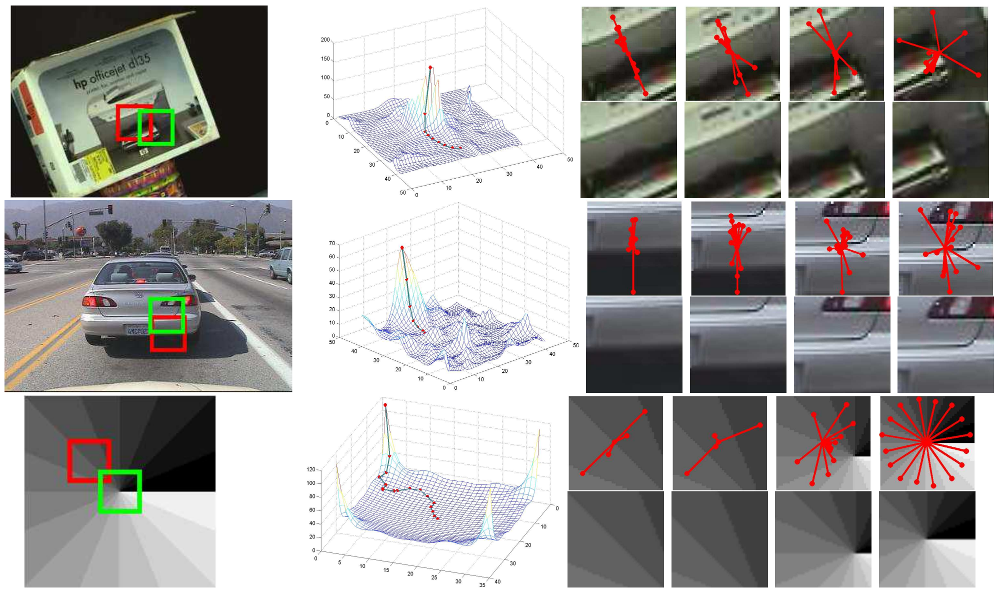


Figure 4.1.3: column (a): the red rectangle indicates the start kernel position, the green one is the optimized kernel placement found by the gradient-based searching algorithm. column (b): the corresponding descending value of the κ_S. column (c): the regions covered by the kernel, which is moving along the direction of the gradient towards the good placement. The red line with a spot indicates the center of mass of each color component.

be simple.

The first strategy can be addressed by the proposed method, that is, the gradient-based searching algorithm can find the good placements near the initialized ones.

The second strategy states that those multiple kernels are expected to maintain an invariant structure, thus serving as an consistent description of the object with good fidelity but great simplicity.

If a kernel is good for tracking at the beginning, but is not suitable for tracking due to view or illumination changes afterwards, this kernel is considered unstable and should be pruned away. It is also required that the stable structure be simple, although the object could be complex. By this means, we can adopt the scheme of multiple collaborative kernel tracking in Chapter 3 [13], which has superior kernel observability than single kernel, to coordinate those representative kernels for an overall reliable tracking performance

However, there is no general answer to the question that what kind of structure of multiple kernels should be chosen, and this is in fact an ongoing research topic in computer vision [12]. For simplicity, we chose triangle, which is easy to manipulate and works well in many sequences.

For each triangle, we build a 2D histogram, recording the 2 internal angles of that triangle. We obtain the statistics of this 2D histogram over a training sequence. The most stable triangle, with all the internal angles exhibiting little variation, is expected to yield a peak in this 2D histogram. So, for each triangle formed by 3 kernels, we measure the entropy of its associated 2D histogram, and choose the one, which has the minimum entropy, to be the most stable triangle kernel structure.

Then, we use this triangle modelling for collaborative kernel tracking [13]. This optimal multiple kernel placement and the mining for stable structure are fully automatic after the initialization of a set of kernels at the very beginning. The initialization can be done either manually or evenly on the image grid.

4.3 Scale-invariant Kernel Placement

A placement is good for kernel if the condition number evaluated on that region is small. This placement is even better if its associated condition number achieves a local minimum, i.e., it is the only one that should be selected from its neighborhood. However, it is obvious that the condition number changes when the scale of the kernel changes, therefore, the local minima property could also vary w.r.t the scale.

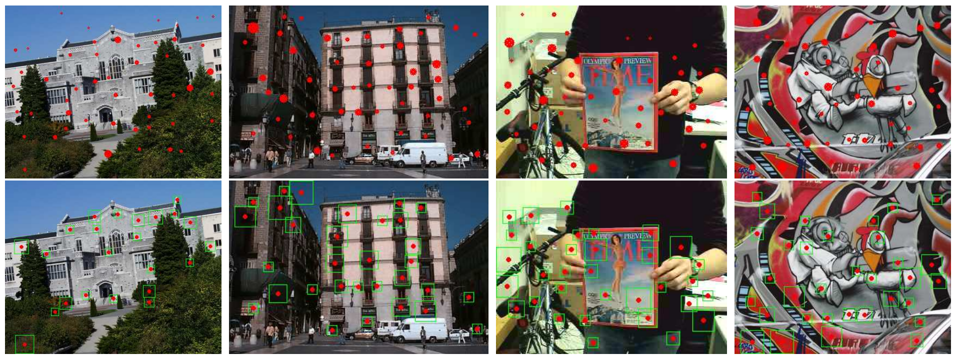
An interesting question is that *if there exists a placement of a kernel, whose condition number is the local minima for all, or for a large number of different, scales?* and how to find them?

Placing a kernel at such a place is invariant to the scale changes, meaning that, it can yield reliable tracking performance when changes occur in the kernel scale, or in the image scale, or in the scales of both image and kernel, since scale changes in the image and in the kernel is just a relative term. This is in fact a very nice property.

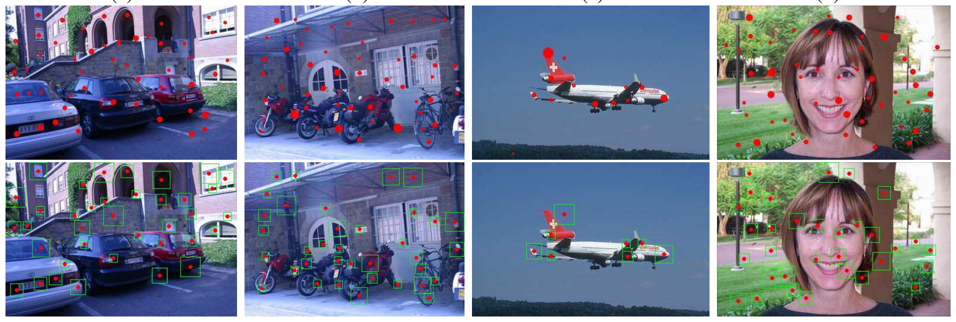
In order to find such placements, rather than brute force evaluating the condition number through all the scales and neighborhood, the searching algorithm in Sec 4.1.4 offers considerable efficiency.

We can evenly initialize a set of kernel samples with different scales on the grid. Then, let these kernels converge to their corresponding local minima. When all the kernel samples are converged, a distribution of the places of converged kernels are obtained. The set of local maxima of such a distribution indicate the set of scale-invariant kernel placements.

In experiment, we generalize kernels with 7 different scales, each scale with 400 evenly initialized kernel samples. The places, to which a large amount of kernel samples are converged, are shown in the top rows of Fig. 4.3.1 (a)-(h). It can be seen that most of these places are featured by a region with diversified color pixels' intensity and spatial distributions.



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



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

Figure 4.3.1: In each figure (a)-(h). Top row: scale-invariant kernel placement. Larger circle means higher density of converged kernel samples. Bottom row: good kernel regions with appropriate scale selection after pruning texture regions.

4.3.1 Scale selection

With scale-invariant kernel positions in hand, we can further proceed one step. That is, to choose the *appropriate scale* of the kernel for that region. Although the scale-invariant region yields the local minimum of κ_S in most of the scales within its neighborhood, different scales still have different κ_S, and of course, the scale associated with a lower κ_S is more preferred. In the following, denote $s_i, i = 1, 2, \cdots$ the discretized scales with increasing order.

One criterion is to choose the scale with a low κ_S, that is to maximize

$$\mathcal{Q}(s_i, \mathbf{c}) \stackrel{\triangle}{=} \frac{1}{\kappa((\mathbf{M}_{s_i,\mathbf{c}})^T(\mathbf{M}_{s_i,\mathbf{c}}))},$$

where $\mathbf{M}_{s_i,\mathbf{c}}$ is obtained at **c** with scale s_i.

Another criterion should also be considered, that is to maximize the difference between the histograms, **q**, obtained at the current scale s_i and the immediately adjacent larger scale s_{i+1}. This implies that the object of interest, kernel of scale s_i, should differ from its adjacent background, the region of scale s_{i+1}, to some large extent. Denote

$$\mathcal{W}(s_i, \mathbf{c}) \stackrel{\triangle}{=} d(\mathbf{q}_{s_i,\mathbf{c}}, \mathbf{q}_{s_{i+1},\mathbf{c}})$$

By combining the above two functions, the appropriate scale at position **c** is determined as

$$s^* = \arg\max_{s_i}(\mathcal{Q}(s_i, \mathbf{c}) \cdot \mathcal{W}(s_i, \mathbf{c}))$$

The attractive properties of this approach is that, $\mathcal{Q}(s_i, \mathbf{c})$ ensures that the tracking algorithm can be stable on such a scale, while $\mathcal{W}(s_i, \mathbf{c})$ tells the fact that there is sufficient difference, or say, discriminance, between the object and the background, thus preventing the tracking algorithm being distracted by the background.

4.3.2 Pruning text regions

Here, another issue needed to be considered is the texture region, which is the major source of confusion in determining the regions' "goodness" by analyzing eigenvalues, or in essence, singular values, because texture region can also yield comparative eigenvalues but is in general not the region of interest. To deal with this problem, in [28], a threshold is set to ensure that the minimal eigenvalue is large enough. Here, we can use the S-norm to achieve the equivalent effect of pruning away texture regions, but with a much more efficient *closed-form* formulation.

As suggested in [26], $\|(\mathbf{M}^T\mathbf{M})^{-1}\|_S = \frac{1}{\sigma_1} + \frac{1}{\sigma_2} = \frac{\sigma_1+\sigma_2}{\sigma_1\sigma_2}$, and $\frac{1}{\sigma_2} \leq \frac{1}{\sigma_1} + \frac{1}{\sigma_2} \leq \frac{2}{\sigma_2}$

So, in order to get a large σ_2, we should select those regions which can yield a small $\frac{\sigma_1+\sigma_2}{\sigma_1\sigma_2}$, and this can be computed explicitly as

$$\frac{\sigma_1 + \sigma_2}{\sigma_1 \sigma_2} = \frac{\sum (d_x^j)^2 + \sum (d_y^j)^2}{\sum (d_x^j)^2 \sum (d_y^j)^2 - (\sum (d_x^j d_y^j))^2} \tag{4.3.1}$$

So, after we find those scale-invariant kernel regions (Sec. 4.3), choose the appropriate scale (Sec. 4.3.1), we can further filter out texture regions, which yield large values of Eq.(4.3.1).

The result is shown in the bottom rows of Fig. 4.3.1 (a)-(h). It can be seen that, in the top rows, there are some detected good regions, denoted as red points, located at texture regions, such as in the sky, or in the bush etc., while they are successfully pruned away as shown in the bottom rows.

The property of such placement for featured region selection, pattern recognition will be our main future work.

4.4 Discussions

4.4.1 Region selection vs. feature point selection

It may be noted that the form of $\mathbf{M}\Delta\mathbf{c} = \sqrt{\mathbf{q}} - \sqrt{\mathbf{p}(\mathbf{c})}$ is similar to that of feature point matching [26][28], in that they are all in the general form of solving a linear equation, i.e., **Ax=b**. Actually, the property of the matrix $\mathbf{A}$ has been actively studied in the past decades for point matching, such as computing the optical flow [5][24][25][28]. Some work also extends the point matching framework to address other geometrical features such as lines [20][32].

However, our work is different from those work in the following aspects.

1) The content of matrix $\mathbf{A}$, denoted as $\mathbf{A}_{region}$ in our work, and $\mathbf{A}_{point}$ in [26][28].

$$\mathbf{A_{region}} = \mathbf{M} = \begin{bmatrix} d_x^1 & d_y^1 \\ \vdots & \vdots \\ d_x^m & d_y^m \end{bmatrix}, \qquad \mathbf{A_{point}} = \begin{bmatrix} g_x^1 & g_y^1 \\ \vdots & \vdots \\ g_x^n & g_y^n \end{bmatrix}.$$

where $[d_x^j \quad d_y^j]$ is the center of mass of all pixels of color j, for m color bins, $j = 1, ..., m$, and $[g_x^i \quad g_y^i]$ being the image gradient of pixel i w.r.t x and y axes, for n pixels within a small window around the feature point, $i = 1, ..., n$.

The different content determines that their analytical results are different. That is, there is no certain correspondences between optimal feature points and optimal regions for tracking. We think these works have different practical impacts in real applications.

2) The criterion and the analytical method In our work, the criterion is $\kappa_S(\mathbf{M}^T\mathbf{M})$, which has the equivalent effect as analyzing $\kappa_2(\mathbf{M}^T\mathbf{M})$. In [28], the criterion is that the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ is larger than a threshold. In

[26], $\|(\mathbf{A}^T\mathbf{A})^{-1}\|_S$ is checked, which has been shown to have the same effect as requiring the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$ to be larger than a threshold. In contrast, our criterion takes into account of both singular values of **M**, which is more general.

As for the analytical method, we have shown that κ_S is equivalent to κ_2 when considering the 2×2 covariance matrix case, which leads to some nice analytical properties.

3) Gradient descent search We provide a gradient based searching scheme to find the optimal regions within an image efficiently, which avoids exhaustive search. But the selection of feature point has to be exhaustive.

4.4.2 Interpretation of condition number criterion using the S-norm

We already know the structure of $\mathbf{M} = [\mathbf{v_x} \quad \mathbf{v_y}]$, where $\mathbf{v_x} = [d_x^1, \ldots, d_x^m]^T$, $\mathbf{v_y} = [d_y^1, \ldots, d_y^m]^T$ are the X-coordinates and Y-coordinates of the the centers of color masses, respectively. Then we have

$$\sum_j (d_x^j)^2 = \|\mathbf{v_x}\|^2, \quad \sum_j (d_y^j)^2 = \|\mathbf{v_y}\|^2,$$

$$(\sum_j (d_x^j d_y^j))^2 = \|\mathbf{v_x}^T\mathbf{v_y}\|^2 = \|\mathbf{v_x}\|^2\|\mathbf{v_y}\|^2\cos^2(\theta).$$

where θ is the angle between vectors $\mathbf{v_x}$ and $\mathbf{v_y}$.

Recall Eq.(4.1.7),

$$\begin{aligned}\kappa_S(\mathbf{M}^T\mathbf{M}) &= \frac{(\|\mathbf{v_x}\|^2+\|\mathbf{v_y}\|^2)^2}{\|\mathbf{v_x}\|^2\|\mathbf{v_y}\|^2-\|\mathbf{v_x}\|^2\|\mathbf{v_y}\|^2\cos^2(\theta)} \\ &= \frac{\frac{\|\mathbf{v_x}\|^4+\|\mathbf{v_y}\|^4}{\|\mathbf{v_x}\|^2\|\mathbf{v_y}\|^2}+2}{1-\cos^2(\theta)} \geq \frac{4}{1-\cos^2(\theta)} \geq 4.\end{aligned}$$

It is easy to verify that the desirable minimum of the above condition number is achieved when

$$\|\mathbf{v_x}\| = \|\mathbf{v_y}\|, \quad \text{and} \quad cos(\theta) = 0, \text{i.e.,} \mathbf{v_x} \perp \mathbf{v_y},$$

which implies that the roles of X and Y coordinates of those centers of color masses are equivalent and interchangeable. The optimal case would be that these mass centers are located symmetrically around the center of kernel **c**. This is actually the same as the interpretation we have given in Sec. 4.1.2, but from another point of view. A perfect example is already shown in the left most column of Fig. 4.1.1.

4.5 Experiment

In this section, experiments using real video sequences demonstrate the effectiveness and usefulness of the proposed algorithm for efficient optimal kernel placement.

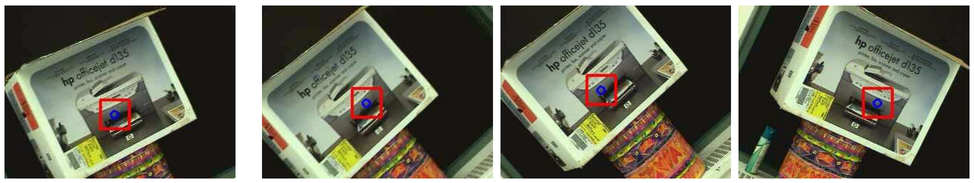
4.5.1 Single kernel

For an object of interest, arbitrarily labelling a region, which meets some special standards, such as owning a high variance, strong edges, or a high entropy, and assigning a kernel on it, may not be optimal. In many cases, unexpected tracking failures make people change the kernel placement through trial and error.

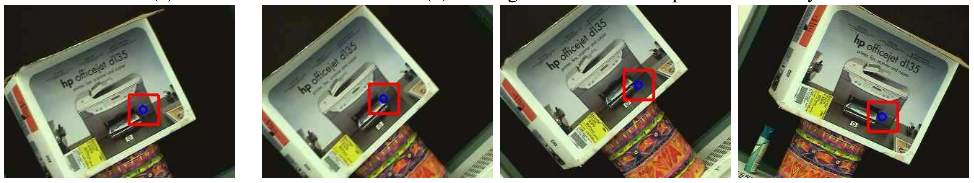
On the contrary, placing kernels by the criterion presented in Sec. 4.1.3 and the efficient searching algorithm derived in Sec. 4.1.4 can easily bring more reliable tracking performance.

Fig. 4.5.1(b) shows the tracking result with a kernel initialized as in Fig. 4.5.1(a). The tracking is not stable, since the unidirectional color distribution in the initial place yields a large condition number. Using the same initialization as (a), we apply the *gradient-based algorithm* and find a good kernel placement, with much lower condition number, as shown in (c), the corresponding tracking result is shown in (d). More reliable performance is obtained.

Fig. 4.5.2(a) shows an arbitrarily initialized kernel placement, the corresponding tracking result is in Fig. 4.5.2(b), in which drifting is observed. In contrast, the searching algorithm moves the place from (a) to a good placement as in (c). The tracking performance is instantly improved a lot, as shown in (d). Notice



(a) (b)tracking result of a kernel placed arbitrarily.



(c) (d) tracking result of a kernel with initial location optimized.

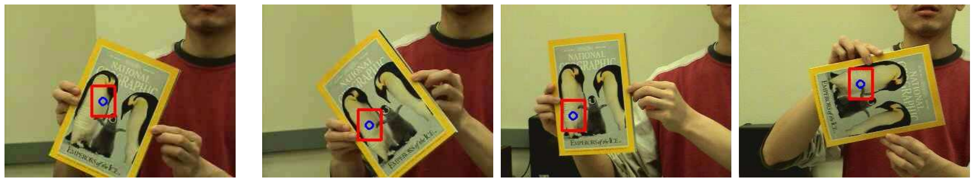
Figure 4.5.1: Tracking with (bottom row) and without (top row) kernel placement optimization.

that the location in Fig. 4.5.2(a) and Fig. 4.5.2(c) is very close, and this is indeed difficult for manual initialization, as heedlessly marking a region, to achieve a robust performance. This shows that the searching algorithm effectively helps us to discriminate good and bad regions for tracking.

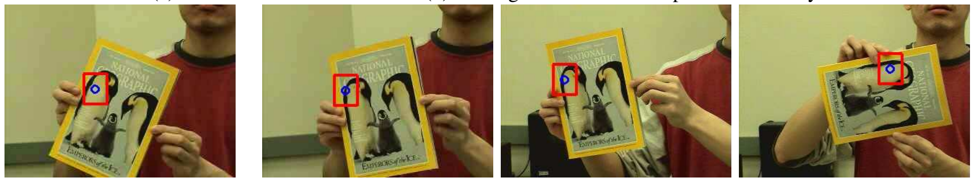
4.5.2 Multiple collaborative kernels

Good strategies to place multiple kernels are that I) each kernel is at a good placement, II) the structure of the multiple kernels should be stable.

To track a region of interest, we initialize a set of kernels on the grid, and run the searching algorithm to find good placements. By this means, we can have a



(a) (b)tracking result of a kernel placed arbitrarily.



(c) (d) tracking result of a kernel with initial location optimized.

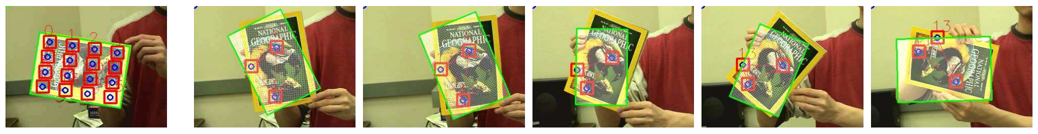
Figure 4.5.2: Tracking with (bottom row) and without(top row) kernel placement optimization.

large coverage of the possible multiple kernel combinations and safely avoid the risk of having bad manually labelled regions.

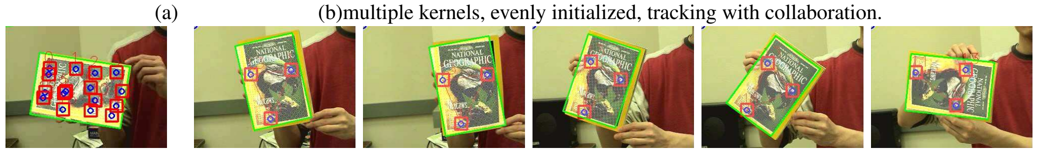
Then we track these kernels over a training sequence, which can be the first several frames of the video. We choose the most stable triangle formed by 3 kernels.

In Fig. 4.5.3(a), multiple kernels are evenly initialized on the grid within the region of interest. *Without* the gradient searching algorithm, that is, we just accept and start from the initial kernel locations, mine for the most stable triangle and apply collaborative tracking scheme. The result is shown in Fig. 4.5.3(b).

Fig. 4.5.3(c) shows the good kernel placement found by applying the gradient searching algorithm on the kernel locations in (a), the corresponding tracking result is shown in (d). In all the figures, the bounding box of the object of interest is reconstructed by conferring the initial localization of the kernels w.r.t. the object. How well can we know the position and orientation of the original object measures the quality of collaborative tracking. It is seen that the performance of kernels, whose locations are optimized, is much better. This is because the kernels, without placement optimization, are more likely to have a large condition number and thus having more exposure to the unstableness in the tracking.



(a) (b)multiple kernels, evenly initialized, tracking with collaboration.

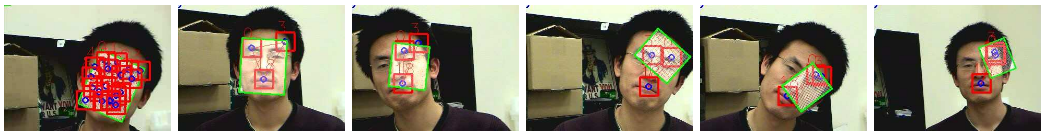


(c) (d) multiple kernels, location optimized, tracking with collaboration.

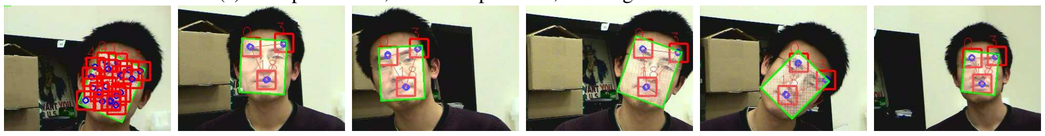
Figure 4.5.3: Multiple kernel tracking, with (bottom row) and without (top row) placement optimization.

The next three sequences involve object scale changes, in which the optimal multiple collaborative kernel tracking works well.

Fig. 4.5.4(a) and Fig. 4.5.4(b) show the tracking result from the same initial kernel locations (left most column), after searching from the evenly gridded initialization. *Without* collaborative scheme applied, the unsatisfactory result in Fig. 4.5.4(a) shows that, although all the kernels are good at the beginning, they still can lost due to various disturbances in tracking, such as, unexpected abrupt motions, disturbance from the object with similar appearance or illumination changes. Since the kernels track independently, they cannot recover themselves, such that the face is lost track. By collaborating the initially good kernels, the result of localizing the face by the 3 kernels is more reliable, as shown in Fig. 4.5.4(b).



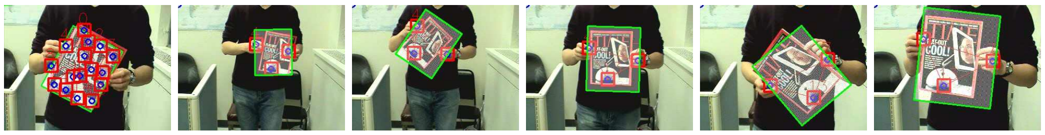
(a) multiple kernels, location optimized, tracking without collaboration.



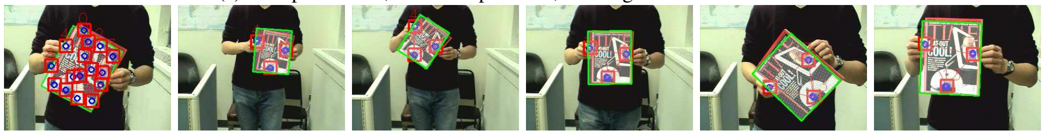
(b) multiple kernels, location optimized, tracking with collaboration.

Figure 4.5.4: Multiple kernel tracking, with (bottom row) and without (top row) collaboration.

Fig. 4.5.5 and Fig. 4.5.6 have the similar settings as in Fig. 4.5.4. The result with collaboration, row (b), again yields much more reliable reconstruction performance of estimating the position, orientation and the scale of the magazine cover.



(a) multiple kernels, location optimized, tracking without collaboration.

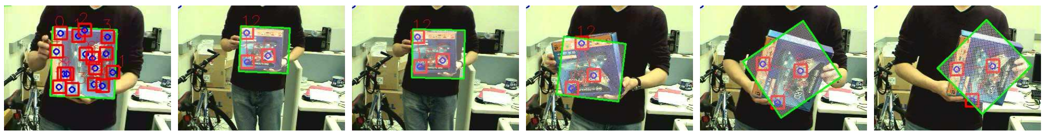


(b) multiple kernels, location optimized, tracking with collaboration.

Figure 4.5.5: Multiple kernel tracking, with (bottom row) and without (top row) collaboration.

4.6 Remarks

To summarize, in this chapter, we present a detailed analysis to the criterion of optimal kernel placement. An equivalent criterion is also derived, which has a closed-form representation and enables a nice gradient-based algorithm to find optimal kernel placement efficiently. Placement of temporal-stable multiple ker-


(a) multiple kernels, location optimized, tracking without collaboration.

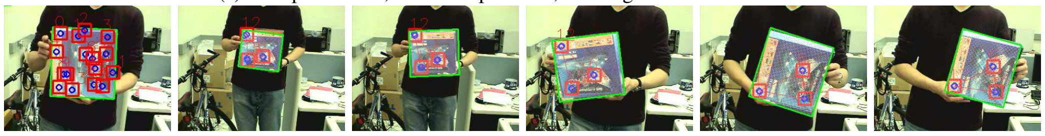

(b) multiple kernels, location optimized, tracking with collaboration.

Figure 4.5.6: Multiple kernel tracking, with (bottom row) and without (top row) collaboration.

nels and scale-invariant kernels are also studied.

Chapter 5

Conclusions

The purpose of this work is to cure the two major and frequently encountered singularities in kernel based tracking, which are concerned with kernel observability and tracking stability. The contributions include (1) the theoretical results that unify the study of the motion observability issue in most kernel-based methods including single and multiple kernels; (2) a principled way of designing observable kernels, i.e.. the multiple collaborative kernels, that can be easily generalized to complex objects and motions; (3) an efficient computational paradigm to cope with complex objects and motions due to the "collaboration" among a set of inter-correlated kernels, each of which only takes charge of recovering a simpler motion; (4) a closed-form criterion for choosing the optimal kernel placement, on which a much more reliable tracking performance can be achieved; (5) a gradient-based searching algorithm to find such optimal kernel placements, which greatly reduces the computational cost compared with the commonly used exhaustive searching. These new theoretical results and new algorithms help us to better understand and implement the kernel-based tracking method.

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth, “Learning to detect objects in images via a sparse, part-based representation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004, vol. 26, no. 11, pp. 1475-1490.

[2] J. Benesty and T. Gansler, “A recursive estimation of the condition number in the RLS algorithm”, *Proc. IEEE Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 25-28.

[3] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto, “Recognition of human gaits”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001, vol. 2, pp. 52-57.

[4] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, London, 1998.

[5] M. Brooks, W. Chojnacki, D. Gaeley, and A. Hengel, “What value covariance information in estimating vision paramaters”, *Proc. International Conference on Computer Vision*, 2001, pp. 302-308.

[6] Y. Cheng, “Mean Shift, Mode Seeking, and Clustering”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995, vol. 17, no. 8, pp. 790-799.

[7] R. T. Collins, “Mean-shift blob tracking through scale space”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 234-240.

[8] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, vol. 24, no. 5, pp. 603-619.

[9] D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2000, vol. 2, pp. 142-149.

[10] D. Comaniciu, V. Ramesh, and P. Meer, “The variable bandwidth mean shift and data-driven scale selection”, *Proc. International Conference on Com-*

puter Vision, 2001, vol. 1, pp. 438-445.

[11] T. F. Cootes, G. J. Edwards, and C. J. Taylor. “Active appearance models”, *Proc. European Conference on Computer Vision*, 1998, pp. 484-498.

[12] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, “Spatial priors for part-based recognition using statistical models”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 10-17.

[13] Z. Fan, Y. Wu, and M. Yang, “Multiple Collaborative Kernel Tracking”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[14] K. Fukunaga, and L. D. Hostetler, “The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition”, *IEEE Transactions on Information Theory*, 1975, vol 21, pp. 32-40.

[15] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990, Second Edition.

[16] G.H. Golub and C.F. Van Loan, “*Matrix computations*”, 2nd edition, The John Hopkins University Press, Baltimore, MD, 1989.

[17] G. D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2004, vol. 1, pp. 790-797.

[18] B. Heisele, T. Serre, M. Pontil, and T. Poggiom "Component-based face detection", *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. 657-662.

[19] R.A. Horn and C.R. Johnson, "*Topics in matrix analysis*", Cambridge, Mass.: MIT Press, 1991.

[20] M. Irani and P. Anandan, "Robust multi-sensor image alignment", *Proc. International Conference on Computer Vision*, 1998, pp. 959-966.

[21] M. Irani, "Multi-frame correspondence estimation using subspace constraints", *Int'l. J. Computer Vision*, vol. 48, no. 3, pp. 173-194, 2002.

[22] M. Isard and A. Blake, "CONDENSATION – conditional density propagation for visual tracking", *Int'l J. Computer Vision*, 1998, vol. 29, pp. 5-28.

[23] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector", *ECCV*, 2004.

[24] Y. Kanazawa and K. Kanatani, “Do we really have to consider covariance matrices for image features?” *Proc. International Conference on Computer Vision*, 2001, pp. 301-306

[25] J.K. Kearney, W.B. Thompson, and D.L. Boley, “Optical flow estimation: an error analysis of gradient-based methods with local optimization”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, vol. 9, pp. 229-244.

[26] C.S. Kenney, B.S. Manjunath, M. Zuliani, G.A. Hewer, and A.V. Nevel, “A condition number for point matching with application to registration and postregistration error estimation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003, vol. 25, pp. 1437-1454.

[27] A. Safonova, J. K. Hodgins, and N. S. Pollard, “Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces”, *Proc. SIGGRAPH*, 2004.

[28] J. Shi and C. Tomasi, “Good features to track”, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593-600.

[29] M. Singh, and N. Ahuja, “Regression based bandwidth selection for segmentation using parzen windows”, *Proc. IEEE International Conference on Computer Vision*, 2003, pp. 2-9.

[30] M. P. Wand and M. C. Jones, *Kernel Smoothing*, Chapman and Hall, 1995, First edition.

[31] J. Wang, B. Thiesson, Y. Xu, and M. F. Cohen, “Image and video segmentation by anisotropic kernel mean shift”, *Proc. European Conference on Computer Vision*, 2004.

[32] R. Wildes, D. Horvonen, S. Hsu, R. Kumar, W. Lehman, B. Matei, and W. Zhao, “Video georegistration: algorithm and quantitative evaluation”, *Proc. International Conference on Computer Vision*, 2001, pp. 343-350.

[33] Y. Wu, G. Hua, and T. Yu, “Tracking Articulated Body by Dynamic Markov Network”, *Proc. International Conference on Computer Vision*, 2003, pp. 1094-1101.