A

Dissertation

On

TRAINING FREE SALIENT OBJECT DETECTION IN VIDEO USING LOCAL STEERING KERNEL

Submitted in Partial fulfillment of the requirement

For the award of the degree of

MASTER OF TECHNOLOGY

In

(Signal Processing and Digital Design)

Submitted by

Yogesh Kumar

Roll No. 08/SPD/2K10

Under the Guidance of

Asst. Prof. D. K. Vishwakarma

Department of Electronics and Communication Engineering

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

BAWANA ROAD, DELHI- 110042

June 2012

DECLARATION BY THE CANDIDATE

June 2012

Date: ___________

I hereby declare that the work presented in this dissertation entitled **"Training free salient object detection in video using local steering kernel"** has been carried out by me under the guidance of Mr. D. K. Vishwakarma Assistant Professor, Department of Electronics & Communication Engineering, Delhi Technological University, Delhi and hereby submitted for the partial fulfillment for the award of degree of Master of Technology in Signal Processing & Digital Design at Electronics & Communication Department, Delhi Technological University, Delhi.

I further undertake that the work embodied in this major project has not been submitted for the award of any other degree elsewhere.

Yogesh Kumar
DTU/M.Tech/176
M.Tech (SP&DD)

CERTIFICATE

It is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

Mr. D. K. Vishwakarma
Assistant Professor
Electronics & Communication Department
Delhi Technological University, Delhi-42

Dated:——————

ACKNOWLEDGEMENTS

At this point I would like to thank the people that helped me producing this dissertation. First, I thank **Dr. Rajiv Kapoor** Head of Department (Electronics and Communication Engineering, DTU), and **Mr. D. K. Vishwakarma** for giving me the opportunity to write this dissertation and supporting me along the way. Next, I would like to say thanks to all my seniors and friends for their goodwill and support that helped me a lot in successful completion of this dissertation.

Yogesh Kumar
DTU/M.Tech/176
M.Tech (SP&DD)

Table of Contents

Certificate ii

Acknowledgment iii

Table of contents iv

List of Figures vii

List of Tables viii

Abstract 1

1 **Introduction** ……………………………………2

1.1 Background of the Proposed Approach ……………3

2 **Kernel Regression** ……………………………………6

2.1 Data Adaptive Regression Kernel ……………………6

2.1.1 Kernel Regression in 1-D ……………………………6

2.1.2 Kernel Regression in 2-D ……………………………10

2.2 Smoothing Matrix Selection ……………………………14

3 **Data Adaptive Kernel Regression** ……………………16

3.1 Data Adaptive Kernels ……………………………………16

3.1.1 Bilateral Kernels ……………………………………17

3.1.2 Steering Kernels ……………………………………20

4 **Principal Component Analysis** ……………………27

4.1 Principal Component Analysis …………………………28

4.1.1 Dual PCA ……………………………………………31

4.2 Kernel PCA ……………………………………………32

4.2.1 Centering ……………………………………………33

4.3 Proof of PCA ……………………………………………34

5 **Resemblance Map and Significance Testing** …………36

5.1 Feature Representation …………………………………36

5.2 Matrix Cosine as a Measure of Similaritry ………………36

5.3 Non-Parametric Significance Test and Non-Maxima Suppression ………………38

6 **Methodology** ……........…..41
6.1 Histogram Intersection …......…........…..41
6.2 Locally Adaptive Kernel Regression …........…......…..42

7 **Simulation and Result** …..47
7.1 Result ….....................…..47
7.2 Limitation …..48
7.3 Conclusions …..49

A Equivalent Kernels 51
B Local Gradient Estimation 53
References 54

List of Figures

1.1) Given a query image Q, we want to detect/localize objects of interest in a target image *T*. *T* is divided into a set of overlapping patches. - - - - - - - - - - - - - - - - - - 4

2.1) The effect of the regression orders: (a) Zeroth order kernel regression (constant model, N= 0), (b) First order kernel regression (linear model, N = 1), and (c) Second order kernel regression (quadratic model, N = 2) - - - - - - - - - - - - - - -9

2.2) The data model for the kernel regression framework in 2-D. - - - - - - - - - - - - - 10

3.1) Kernel contours in a uniformly sampled data set: (a) Kernels in the classic method depend only on the spatial distances, and (b) Data-adaptive kernels elongate with respect to the local edge structure. -17

3.2) Bilateral kernel weights given by (2.2) at flat, edge, and Lena's eye regions of a clean image. -18

3.3) A visual analysis of the bilateral equivalent weight function W_i (K, $\mathbf{H_s}$, h_p, N, $x_i - x$, $y_i - y$) in (3.4) at a variety of image structures; flat, strong edge, corner, texture, and weak edge for the zeroth and second order (N = 0 and 2). - - - - 19

3.4) Horizontal cross-sections of the bilateral equivalent weight function W_i (K, $\mathbf{H_s}$, h_p, N, $x_i - x$, $y_i - y$) at the strong edge for the zeroth and second order (N = 0and 2): (a)-(b) the footprints of W_i for the zeroth and second order, respectively. - 20

3.5) Bilateral kernel weights given by (3.2) at flat, edge, and Lena's eye regions of a noisy image. The noisy image is given by adding white Gaussian noise. - - - - 21

3.6) A schematic representation illustrating the effects of the steering matrix and its components on the size and shape of the regression kernel footprint. - - - - - - - 23

3.7) A schematic representation of the estimates of local covariance metrics and the steering kernel weights at a local region with one dominant orientation:

(a) First, we estimate the gradients and compute the local covariance matrix C_i by (2.13) for each pixel, and

(b) for y_{13}, we compute the steering kernel weights with C_i for neighboring pixels. Even though, in this case, the spatial distances between y_{13} and y_1 and between y_{13} y_{21} are equal, the steering kernel weight for y_{21} (i.e. $K_{H21}(x_{21} - x_{13})$) is larger than the one for y_1 (i.e. $K_{H1}(x_1 - x_{13})$). This is because y_{13} and

y_{21} are located along the same edge. -------------------------------26

5.1) (a) Resemblance map (RM) which consists of $|\rho_i|$

(b) Resemblance map (RM) which consists of $f(\rho_i)$. ------------------ 38

5.2) (a) Query

(b) Target with detection

(c) Two significance tests

(d) Non-maxima suppression ---------------------------------39

6.1) System Overview -- 42

6.2) Steering Kernel -- 43

7.1) Query Image --- 47

7.2) Histogram intersection graphical output showing the frame at which abrupt change occurs. ---47

7.3) Tanks detected in the system in the frames. ------------------------- 48

7.4) Chained query image generation

(a) shows the detection using the image of figure 7.1(a)

(b) shows the detection using the image of figure 7.1(b) ------------- 49

List of Tables

1. Algorithm 4.1 – Direct PCA Algorithm..30

2. Algorithm 4.2 – Dual PCA …..32

3. Algorithm 5.1 – Pseudo-code for training-free video-shot object detection …....45

Abstract

Video is an ample source of information. It provides visual information about scenes, actions or some activity. However, this information is implicitly buried inside the raw video data, and is provided with the cost of very high temporal redundancy. While the standard sequential form of video storage is adequate for viewing in a "movie mode", it fails to support rapid access to information of interest that is required in many of the emerging applications of video. Extracting a small number of key frames that can abstract the content of video is very important for efficient browsing and retrieval in video databases. This paper presents a training-free approach of detection of salient object in a segmented video sequence. A video can be segmented by assuming that there is an abrupt change in the video sequence. After the video is segmented the frame obtained is matched using local adaptive kernel regression (LARK) against a single query.

Chapter 1

Introduction

When we have large database of images and videos in the form of home videos or videos from news feeds from different channels, they are always abundant in information in the form of space time images. If we want to find a particular object then we have to go through the whole video to search for that object. This search could be based on metadata such as color, keyword, captioning or description to the image, but relying on metadata is not satisfactory since it tend to produce large result with most of them being garbage only, and it's almost impossible to manually search for keywords of an image in a video.

Content-based image retrieval (CBIR), is the application of computer vision techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases. Content-based means that the search will analyze the actual contents of the image. Most of the image retrieval techniques are based on the some kind of feature extraction or content based algorithm in which resultant feature vector is compared with the feature vector of the query image. The closest image in comparison with the query image from the feature database is returned such that the result images share common elements with the provided query. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself. A system that can filter images based on their content would provide better indexing and return more accurate results.

Tamura, Mori & Yamawaki (1978) [1] proposed texture based feature in which measures look for visual patterns in images and how they are spatially defined. The relative brightness of pairs of pixels is computed such that degree of contrast, regularity, coarseness and directionality may be estimated. Tushabe and Wilkinson [2] use shape filters to identify given shapes of an image which will often be determined first applying segmentation or edge detection to an image. Recent techniques in image retrieval include boundary detection feature extraction by Dr Kekre H.B, Mishra D, (2011)[3], in the boundary detection method, the binary image is scanned until a boundary is found. Scanning is done on the basis of K- Nearest Neighbor Method. PCA Algorithm for Feature Extraction by Dr Kekre. H.B, Thepade S.D., Maloo A., (2010)[4] a 2-D data is reduced into one-dimensional format by concatenating each row into a long thin vector. The average image i.e the common part of each image in the test data is calculated and subtracted from the original image to get the unique part of the image. Dr Kekre H.B, Mishra D, (2010)[5] present a feature extraction using DCT transform which make use of the DCT transform to design the sectors required for the search and retrieval of images from a database. Feature extraction using fast fourier transform is used to generate the feature vectors based on the mean values of real and imaginary parts of complex numbers of polar coordinates in thefrequency domain

Many CBIR systems have been developed, but the problem of retrieving images on the basis of their pixel content remains largely unsolved and for image retrieval they are mostly dependent upon the training of algorithm of classifier.

Here we present an approachh which uses training free method and finds the object pattern mostly on the basis of pixel conent. In a video indexing system we are required to segment the video and retrieve the salient frames of the video and index them according to our requirement. Image retrieval is the first problem in the task of video indexing. One effective way of approaching this problem would be to segment the video in the frames and detecting the change in frame which could be gradual or abrupt. When this changing frame is obtained we can query this frame for salient object detection to index it using local adaptive regression kernel. There is a recent work on video segmentation using syntactic features with a fuzzy theoretic approach by R.S. Jadon, Santanu Chaudhary and K. K. Biswas [7]. Feature matching algorithm in this paper makes use of the generic object detection algorithm proposed by Seo and Milanfar [8].

1.1 Background of the Proposed Approach

Our proposal to the object detection task are three-fold. First, we propose to use histogram intersection to segment the video in the video shots which comprises of a frame at which an abrupt change in the videos frames has occurred. Second we propose to use local regression kernels as descriptors which capture the underlying local structure of the data exceedingly well, even in the presence of significant distortions. Third, we propose the approach to the detection object using a non-parametric nearest-neighbor classifier, along with a generalization of the cosine similarity to the matrix case. The origin and motivation behind the use of these local kernels is the earlier work on adaptive kernel regression for image processing and reconstruction [8]. In that work, localized nonlinear filters were derived, which adapt themselves to the underlying structure of the image in order to very effectively perform denoising, interpolation, and deblurring [9]. The fundamental component of the so called steering kernel regression method is the calculation of the local steering kernel (LSK) which essentially measures the local similarity of a pixel to its neighbors both geometrically and radiometrically. The key idea is to robustly obtain local data structures by analyzing the radiometric (pixel value) differences based on estimated gradients, and use this structure information to determine the shape and size of a canonical kernel. Denoting the target image (*T*), and the query image (*Q*), we compute a dense set of local steering kernels from each. These

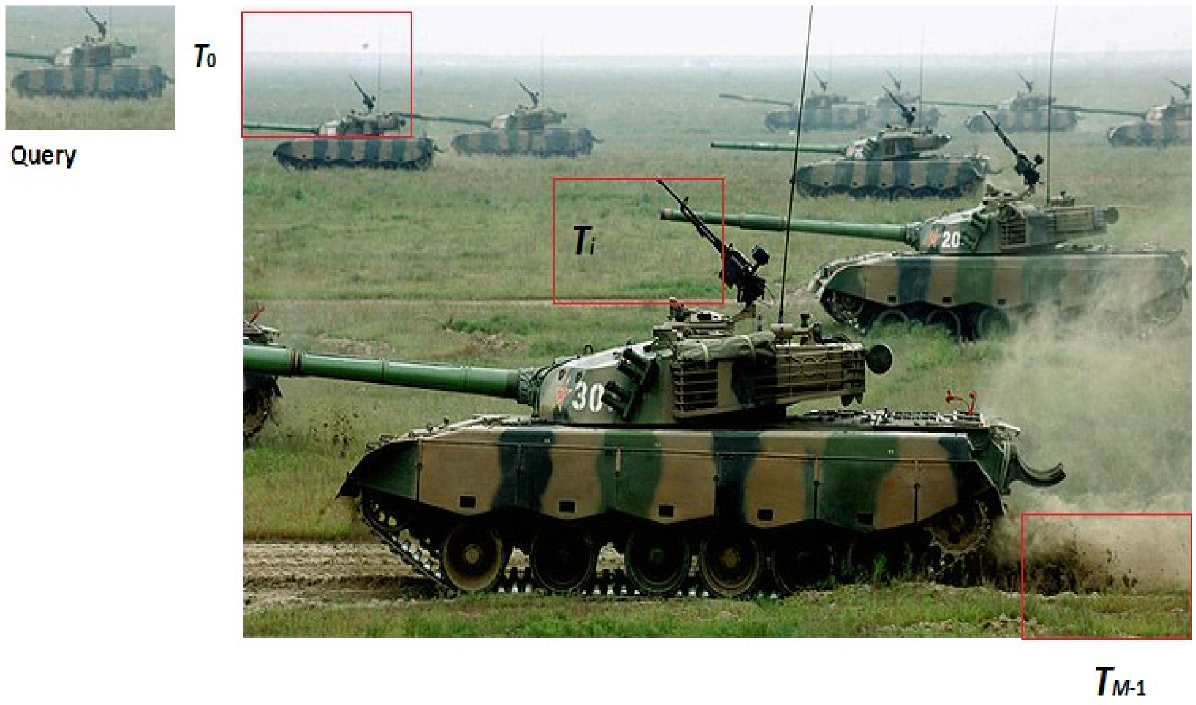


Figure 1.1: Given a query image Q, we want to detect/localize objects of interest in a target image *T*. *T* is divided into a set of overlapping patches.

densely computed descriptors are highly informative, but taken together tend to be redundant. Therefore, we derive features by applying dimensionality reduction using PCA to these resulting arrays, in order to retain only the salient characteristics of the local steering kernels. Generally, T is bigger than the query image Q. Hence, we divide the target image T into a set of overlapping patches which are the same size as Q and assign a class to each patch (T_i). The feature vectors which belong to a patch are thought of as training examples in the corresponding class (See Fig. 1.1). The feature collections from Q and T_i form feature matrices F_Q and F_{Ti}. We compare the feature matrices F_{Ti} and F_Q from i_{th} patch of T and Q to look for matches. We employ "Matrix Cosine Similarity" as a similarity measure which generalizes the cosine similarity between two vectors to the matrix case. We illustrate the optimality properties of the proposed approach using a naive Bayes framework, which leads to the use of the Matrix Cosine Similarity (MCS) measure. In order to deal with the case where the target image may not include any objects of interest or when there are more than one object in the target, we also adopt the idea of a significance test and non-maxima suppression.

Chapter 2

Kernel Regression

Kernel regression is known as a nonparametric approach that requires minimal assumptions, and hence the framework is one of the suitable approaches to the regression problem. In this introductory chapter, we study the classic kernel regression frame-work, and expand it and develop applications for image.

2.1: Classic Kernel Regression and its Properties

In this section, we review the classic kernel regression framework, provide some intuitions on computational efficiency as well as weaknesses of this basic approach, and motivate the development of more powerful regression tools presented in the following chapters.

2.1.1: Kernel Regression in 1-D

Classical parametric image processing methods rely on a specific model of the signal of interest, and seek to compute the parameters of this model in the presence of noise. Examples of this approach range from denoising to upscaling, interpolation, and deblurring. A generative model based upon the estimated parameters is then produced as the best estimate of the underlying signal.

In contrast to the parametric methods, non-parametric methods rely on the data itself to dictate the structure of the model, in which case this implicit model is referred to as *a regression function* [10]. With the relatively recent emergence of machine learning methods, kernel methods have become well-known and used frequently for pattern detection and discrimination problems. Surprisingly, it appears that the corresponding ideas in non-parametric estimation – what we call here kernel regression, are not widely recognized or used in the image and video processing literature. Indeed, in the last decades, several concepts related to the general theory we promote here have been rediscovered in different guises, and presented under different names such as normalized convolution, bilateral filter, edge-directed interpolation, and moving least-squares. To simplify this introductory presentation, we here treat the 1-D case where the measured data are given by

$$y_i = z(x_i) + \varepsilon_i, i = 1,2,\ldots, P \tag{2.1}$$

$z(\cdot)$ is the (hitherto unspecified) regression function, ε_i is the independent and identically distributed zero mean noise value, and P is the number of measured samples in a local analysis window. As such, kernel regression provides a rich mechanism for computing point-wise estimates of the function with minimal assumptions about global signal or noise models.

While the specific form of the regression function $z(\cdot)$ may remain unspecified, if we assume that it is locally smooth to some order N, then in order to estimate the value of the function at any point x given the data, we can rely on a generic local expansion of the function about this point. Specifically, if the position of interest x is near the sample at x_i, we have the N -term Taylor series

$$\begin{aligned} z(x_i) &= z(x) + z'(x)(x_i - x) + \frac{1}{2!} z''(x)(x_i - x)^2 + \ldots + \frac{1}{N!} z^{(N)}(x)(x_i - x)^N \\ &= \beta_0 + \beta_1 (x_i - x) + \beta_2 (x_i - x)^2 + \ldots + \beta_N (x_i - x)^N \end{aligned} \tag{2.2}$$

where $z'(\cdot)$ and $z^{(N)}(\cdot)$ are the first and N-th derivatives of the regression function. The above suggest that if we now think of Taylor series as a local representation of the regression function, estimating the parameters β_0 can yield the desired (local) estimate of the regression function based on the data. Indeed, the parameters $\{\beta_n\}^{N}_{n=1}$ will provide localized information on the nth derivatives of the regression function. Naturally, since this approach is based on local approximation, a logical step to take is to estimate the parameters $\{\beta_n\}^{N}_{n=0}$ from the data while giving the nearby samples higher weight than samples farther away. A least-squares formulation capturing this idea is to solve the following optimization problem:

$$\min_{\beta_n} \sum_{i=1}^{P} \left[y_i - \beta_1 (x_i - x) - \beta_2 (x_i - x)^2 - \cdots - \beta_0 (x_i - x)^N \right] \frac{1}{h} K\left(\frac{x_i - x}{h} \right) \tag{2.3}$$

where $K(\cdot)$ is the kernel function which penalizes distant away from the local position where the approximation is centered, ant the smoothing parameter h (also called the "bandwidth") controls the strength of this penalty. In particular, the function K is a symmetric function which attains its maximum at zero, satisfying

$$\int \delta K(\delta) d\delta, \int \delta^2 K(\delta) d\delta = c \tag{2.4}$$

where c is some constant value. The choice of the particular form of the function K is open, and may be selected as a Gaussian, exponential, or other forms which comply with the above constraints. It is known that for the case of classic regression the choice of the kernel has only a small effect on the accuracy of estimation and therefore preference is given to the differentiable kernels with low computational complexity such as the Gaussian kernel.

Several important points are worth making here. First, the above structure allows for tailoring the estimation problem to the local characteristics of the data, whereas the standard parametric model is generally intended as a more global fit. Second, in the estimation of the local structure, higher weight is given to the nearby data as com-pared to samples that are farther away from the center of the analysis window. Mean-while, this approach does not specifically require the data to follow a regular or equally spaced sampling structure. More specifically, so long as the samples are near the point x the framework is valid. Again this is in contrast to the general parametric approach which generally either does not directly take the location of the data samples into account, or relies on regular sampling over a grid. Third, and no less important the data-adaptive approach is both useful for denoising, and equally viable for interpolation of sampled data at points where no actual samples exist. Given the above observations, the kernel-based methods are well-suited for a wide class of image/video processing problems of practical interest.

Returning to the estimation problem based upon (2.3), one can choose the order N to effect an increasingly more complex local approximation of the signal. In the non-parametric statistics literature, locally constant, linear, and quadratic approximation (corresponding to N = 0, 1, 2) have been considered most widely. In particular, choosing N = 0, a locally linear filter is obtained, which is known as the Nadaraya-Watson Estimator (NWE) [11]. Specifically, this estimator has the form:

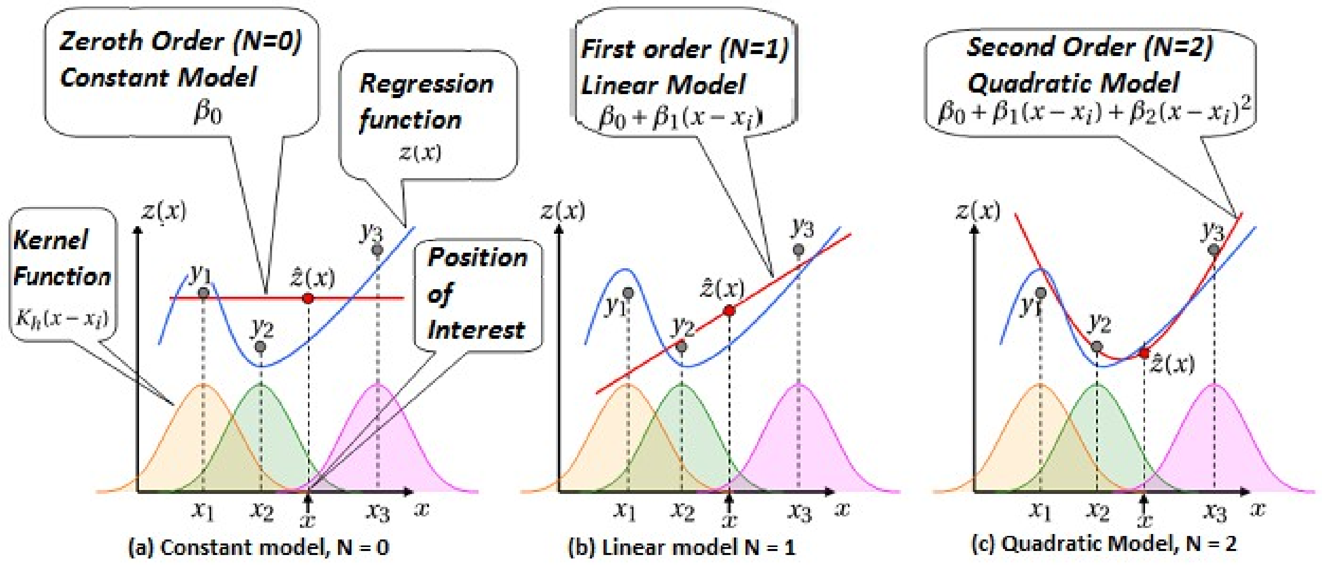
$$\hat{z}(x) = \frac{\Sigma_{i=1}^{P} K_h(x_i - x)\, y_i}{\Sigma_{i=1}^{P} K_h(x_i - x)} \tag{2.5}$$

where

$$K_h(x_i - x) = \frac{1}{h} K\left(\frac{x_i - x}{h}\right) \tag{2.6}$$

The NWE is the simplest manifestation of an adaptive filter resulting from kernel regression framework.

Of course, higher order approximations ($N > 0$) are also possible. The choice of order in parallel with the smoothness (h) affects the bias and variance of the estimate. In general, lower order approximates, such as NWE, result in smoother signals (large bias and small variance) as there are fewer degrees of freedom. On the other hand over-fitting happens in regressions using higher orders of approximation, resulting in small bias and large estimation variance. we illustrate the effect of the regression orders from N = 0 to 2 with a fixed smoothing parameter in Figure 2.1, where the blue curve is the regression (true) function, the gray circles (*y1, y2*, and *y3*) are the measurements with some noise at *x1, x2* and *x3*, respectively, and, in this illustration, we estimate an unknown value at the position of interest x located between the samples *y2* and *y3* with different regression orders. First, for the zeroth order (constant model, $N = 0$), we take only the constant term of Taylor series into account, i.e. β_0. In this case, the kernel regression estimates z(x) by NWE (2.34); a weighted average of nearby samples with the weights that the kernel



Zeroth Order (N=0) Constant Model β_0
Regression function $z(x)$
First order (N=1) Linear Model $\beta_0 + \beta_1(x - x_i)$
Second Order (N=2) Quadratic Model $\beta_0 + \beta_1(x - x_i) + \beta_2(x - x_i)^2$
Kernel Function $K_h(x - x_i)$
Position of Interest
$z(x)$
y_1
y_2
y_3
$\hat{z}(x)$
x_1
x_2
x_3
x
(a) Constant model, N = 0
(b) Linear model N = 1
(c) Quadratic Model, N = 2

Figure 2.1: The effect of the regression orders: (a) Zeroth order kernel regression (constant model, N = 0), (b) First order kernel regression (linear model, N = 1), and (c) Second order kernel regression (quadratic model, N = 2)

function gives. The red circle is the estimated value and the red line is the estimated fitted line.

Similar to the zeroth order case, we draw draw the estimate values and fitted lines by the first and second order kernel regression, which ncorporate up to the linear and quadratic terms, respectively. As seen in Figures 2.1(a)-(c), the estimated line fits the neighboring samples better the higher the regression order becomes. This is because a high order regressor has more degrees of freedom, which is the cause of small bias and large variance. We also note that smaller values for h result in small bias and consequently large variance in estimates.

2.1.2 Kernel Regression in 2-D

Similar to the 1-D case in (2.1), the data model in 2-D is given by

$$y_i = z(x_i) + \varepsilon_i, i = 1, \ldots, P, x_i = [x_{1i}, x_{2i}]^T \tag{2.17}$$

where y_i is a noisy sample at a sampling position x_i (Note: x_{1i} and x_{2i} are spatial coordinates), $z(\cdot)$ is again the (unspecified and bivariate) regression function to be estimated, ε_i is an i.i.d. zero mean noise, and P is the total number of samples in an arbitrary "window" around a position x of interest as shown in Figure 2.2. Correspondingly, the local representation of the regression function using Taylor series (up to the Nth polynomial) is given by

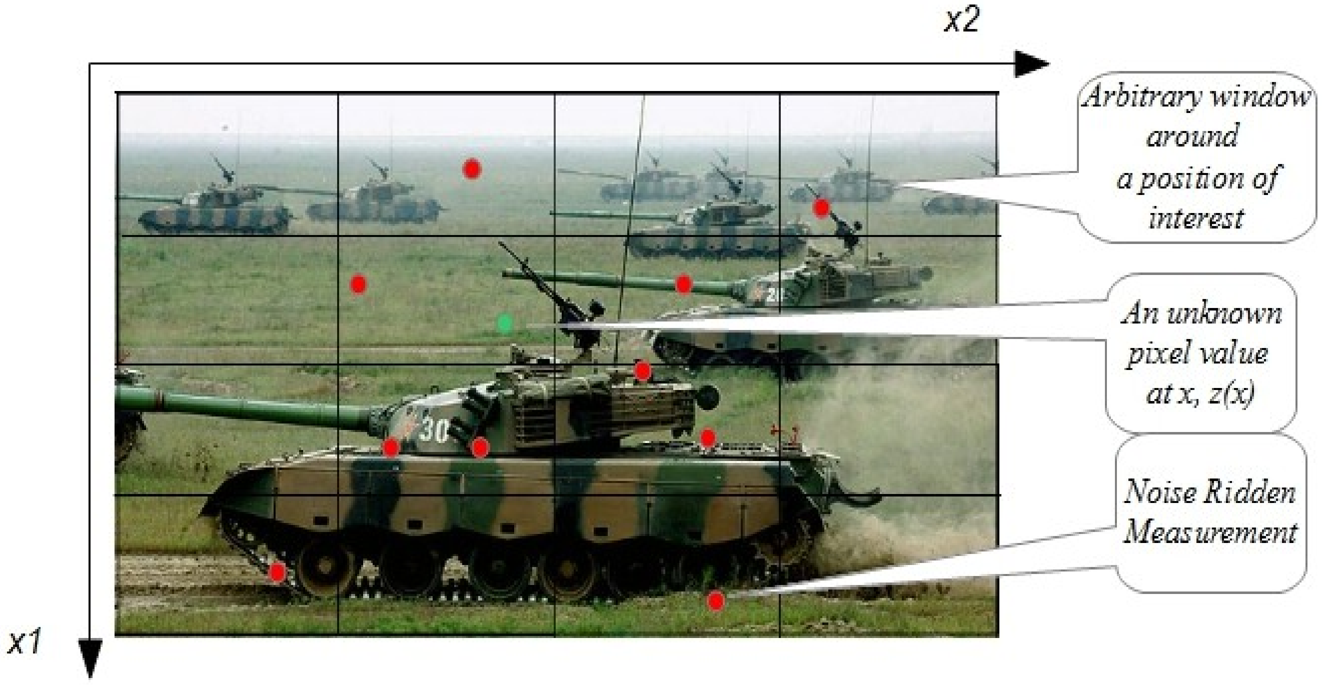

x2
x1
Arbitrary window around a position of interest
An unknown pixel value at x, z(x)
Noise Ridden Measurement

Figure 2.2: The data model for the kernel regression framework in 2-D.

$$z(x_i) \approx z(x) + \{\nabla z(x)\}^T (x_i - x) + \frac{1}{2}(x_i - x)^T \{H\, z(x)\}(x_i - x) + \ldots$$

$$z(x) + \{\nabla z(x)\}^T (x_i - x) + \frac{1}{2} vect^T \{H\, z(x)\}\, vect\{(x_i - x)(x_i - x)^T\} + \ldots \qquad (2.18)$$

where ∇ and *H* are the gradient (2 × 1) and Hessian (2 × 2) operators, respectively, and vect(·) is the vectorization operator, which lexicographically orders a matrix into a column-stack vector. Defining vech(·) as the half-vectorization operator of the "lower-triangular" portion of a symmetric matrix, e.g.,

$$vech\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix}\right) = \begin{bmatrix} a_{11} & a_{12} & a_{22} \end{bmatrix}^T$$

$$vech\left(\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}\right) = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{22} & a_{23} & a_{33} \end{bmatrix}^T \qquad (2.19)$$

and considering the symmetry of the Hessian matrix, the local representation in (2.18) is simplified to

$$z(x_i) \approx \beta_0 + \beta_1^T (x_i - x) + \beta_2^T vech\{(x_i - x)(x_i - x)^T\} + \cdots \qquad (2.20)$$

then, comparison of (2.18) and (2.20) suggests that β_0 is the pixel value of interest, and the vectors β_1 and β_2 are the first and second derivatives, respectively, i.e.,

$$\beta_0 = z(x) \quad , \qquad (2.21)$$

$$\beta_1 = \nabla z(x) = \begin{bmatrix} \frac{\partial z(x)}{\partial x_1} & \frac{\partial z(x)}{\partial x_2} \end{bmatrix}^T \qquad (2.22)$$

$$\beta_2 = \frac{1}{2}\begin{bmatrix} \frac{\partial^2 z(x)}{\partial x_1^2} & 2\frac{\partial^2 z(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 z(x)}{\partial x_2^2} \end{bmatrix}^T \qquad (2.23)$$

As in the case of univariate data, the β_n's are computed from the following optimization problem:

$$min_{\{\beta_n\}} \Sigma_{i=1}^P \left[y_i - \beta_0 - \beta_1^T (x_i - x) - \beta_2^T vech\{(x_i - x)(x_i - x)^T\} - \ldots \right]^2 K_H(x_i - x) \qquad (2.24)$$

with

$$K_H(x_i - x) = \frac{1}{det(H)} K\left(H_{-1}(x_i - x)\right) \qquad (2.25)$$

where *K* is the 2-D realization of the kernel function, and *H* is the 2 × 2 smoothing matrix. For example, if we choose Gaussian function for *K* , the kernel function is expressed as

$$K_H(x_i - x) = \frac{1}{2\pi\sqrt{det(H^T H)}} \exp\{\frac{-1}{2}(x_i - x)^T (H^{TH})^{-1}(x_i - x)\} \tag{2.26}$$

Regardless of the regression order (N) and the dimensionality of the regression function, we can rewrite the optimization problem (2.24) as a weighted least squares optimization problem:

$$\hat{b} = \underset{b}{argmin}\left[(y - Xb)^T K (y - Xb)\right] \tag{2.27}$$

$$\boldsymbol{y} = \begin{bmatrix} y_1 & y_2 & \cdots & y_p \end{bmatrix}^T \tag{2.28}$$

$$\boldsymbol{b} = \begin{bmatrix} \beta_0 & \boldsymbol{\beta}_1^{\boldsymbol{T}} & \cdots & \boldsymbol{\beta}_N^{\boldsymbol{T}} \end{bmatrix}^T \tag{2.29}$$

$$\boldsymbol{k} = diag\left[K_H(x_1 - x) \quad K_H(x_2 - x) \quad \cdots \quad K_H(x_p - x)\right] \tag{2.30}$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & (x_1 - x)^T & vech^T\{(x_1 - x)(x_1 - x)^T\} & \cdots \\ 1 & (x_2 - x)^T & vech^T\{(x_2 - x)(x_2 - x)^T\} & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ 1 & (x_p - x)^T & vech^T\{(x_p - x)(x_p - x)^T\} & \cdots \end{bmatrix} \tag{2.31}$$

with “diag” defining a diagonal matrix. Using the notation above, the optimization (2.24) provides the weighted least square estimator:

$$\hat{\boldsymbol{b}} = \left(x^T KX\right)^{-1} X^T Ky \tag{2.32}$$

Since our primary interest is to compute an estimate of the image (pixel values), the necessary computations are limited to the ones that estimate the parameter β_0. Therefore, the estimator is simplified to

$$\hat{z}(\boldsymbol{x}) = \hat{\beta}_0 = \boldsymbol{e}_1^{\boldsymbol{T}}\left(\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{KX}\right)^{-1} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{Ky} \tag{2.33}$$

where e_1 is a column vector with the first element equal to one, and the rest equal to zero. Of course, there is a fundamental difference between computing β_0 for the $N = 0$ case, and using a high order estimator ($N > 0$) and then effectively discarding all β_n’s except β_0. Unlike the former case, the high regression order method computes estimates of pixel values assuming an N-th order local polynomial structure is present by including higher order polynomial bases as in the matrix X (2.31). Similar to the 1-D case, for $N = 0$, the kernel estimator (2.33) is expressed as

$$\hat{z}(\boldsymbol{x}) = \hat{\beta}_0 = \frac{\sum_{i=1}^{P} K_H(x_i - x)\, y_i}{\sum_{i=1}^{P} K_H(x_i - x)} \tag{2.34}$$

which is Nadaraya-Watson estimator (NWE) in 2-D. For example, when we choose the kernel function as Gaussian (2.26), NWE is nothing but the well-known Gaussian low-pass filter and it provides a pixel value of interest β_0 by a weighted linear combination of the nearby samples. Even the higher order estimator can be generally expressed in the weight linear fashion as

$$\hat{z}(x)=\hat{\beta}_0=\boldsymbol{e}_1^T\hat{\boldsymbol{b}}=\sum_{i=1}^{P}W_i(K,H,N,x_i-x)\,y_i \tag{2.35}$$

where

$$\sum_{i=1}^{P}W_i(.)=1 \tag{2.36}$$

and we call W_i the equivalent kernel function for y_i. It is worth noting that the estimator (2.32) also yield local gradients for the regression orders $N > 0$. While the exact expressions in Appendices A and B yield the mathematical property of the kernel regression estimator, the pixel estimator and the gradient estimator can be simply expressed as follows. We can rewrite the overall estimator (2.32) for the regression order $N > 0$ as

$$\hat{\boldsymbol{b}}=\left(\boldsymbol{X}^T\boldsymbol{KX}\right)^T\boldsymbol{X}^T\boldsymbol{Ky}=W_N\,y=\begin{bmatrix}w_0^T(N)\\ w_1^T(N)\\ w_2^T(N)\\ \vdots\end{bmatrix}\boldsymbol{y} \tag{2.37}$$

where N is the regression order, and w_0(N), w_1 (N), and w_2 (N) $\in R^{P\times 1}$ are the equivalent kernel weight matrices that compute the unknown pixel value and its derivatives as follows. From (2.35), we have

$$\hat{z}(\boldsymbol{x})=\hat{\beta}_0=\boldsymbol{e}_1^T\hat{\boldsymbol{b}}=\boldsymbol{W}_0^T(N)\,\boldsymbol{y} \tag{2.38}$$

$$\nabla\hat{z}(\boldsymbol{x})=\hat{\boldsymbol{\beta}}_1=\begin{bmatrix}e_2^T\\ e_3^T\end{bmatrix}\hat{\boldsymbol{b}}=\begin{bmatrix}w_1^T(N)\\ w_2^T(N)\end{bmatrix}\boldsymbol{y} \tag{2.39}$$

Note that when we estimate the nth derivatives of $z(\cdot)$, the regression order N must be equal or higher than n ($N \geq n$). For instance, $w_1(0)$ and $w_2(0)$ do not exist.

Therefore, regardless of the regression order, the classic kernel regression is local weighted averaging of data (linear filtering), where the order determines the type of complexity of the weighing scheme. This also suggests that higher order regressions ($N > 0$) are equivalents of the zeroth order regression ($N = 0$) but with a more complex equivalent kernel function. In other words, to effect the higher order regressions, the original kernel $K_H(x_i - x)$ is modified to yield a newly adapted “equivalent” kernel.

2.2 Smoothing Matrix Selection

The shape (or contour) of the regression kernel as defined in (2.25), and consequently the performance of the estimator depends on the choice of the smoothing matrix **H**. For the bivariate regression problem, the smoothing matrix **H** is 2 × 2, and it should extend the support of the regression kernel to contain “enough” samples. As illustrated in Figure 2.10, it is reasonable to use a smaller support size in the area with more available samples, whereas a larger support size is more suitable for the more sparsely sampled area of the image. The cross validation “leave-one-out” method is a popular technique for estimating the elements of the local smoothing matrices $\mathbf{H}_i$ for all the given samples y_i. However, as the cross validation method is computationally very expensive, we can use a simplified and computationally more efficient model of the smoothing matrix as

$$\boldsymbol{H}_i = h \mu_i \boldsymbol{I} \tag{2.40}$$

where μ_0 is a scalar that captures the local density of samples (nominally set to $\mu_0 = 1$) and h is the global smoothing parameter.

The global smoothing parameter is directly computed from the cross validation method, by minimizing the following cost function

$$C_{cv}(h) = \frac{1}{P} \sum_{i=1}^{P} \left\{ \hat{z_{i-}}(x_i) - y_i \right\}^2 \tag{2.41}$$

where $\hat{z}_i - (x_i)$ is the estimated pixel values without including the i_{th} sample (i.e., y_0) at x_i. To further reduce the computations, rather than leaving a single sample out, we can leave out a set of samples (a whole row or column).

The local density parameter μ_i is estimated as follows

$$\mu_i = \left\{ \frac{\hat{f}(x_i)}{\exp\left(\frac{1}{P} \Sigma_{i=1}^{P} \log \hat{f}(x_i) \right)} \right\}^{-\zeta} \tag{2.42}$$

where the sample density $\hat{f}(x_i)$ is measured as

$$\hat{f}(x_i) = \frac{1}{P} \sum_{i=1}^{P} K_{H_i}(x_i - x) \tag{2.43}$$

and ζ, the density sensitivity, is a scalar satisfying $0 < \zeta \leq 1$. Note that H_i and μ_i are estimated in an iterative fashion. In the first iteration, μ_i is initialized by 1 and we estimate the density by (2.43). Then, we update μ_i by (2.42) with the estimated density and estimate the density. The process is repeated until μ_i converges (typically, only a few iterations (at most 5 iterations)).

Chapter 3

Data-Adaptive Kernel Regression

In the previous chapter, we studied the classic kernel regression framework and its properties. One fundamental improvement on the above method can be realized by noting that, the local polynomial kernel regression estimates, independent of the regression order N, are always local “linear” combinations of the data. As such, though elegant, relativity easy to analyze, and with attractive asymptotic properties, they suffer from an inherent limitation due to this local linear action on the data. In what follows, we discuss extensions of the kernel regression method that enable this structure to have nonlinear, more effective, action on the given data: data-adaptive kernel regression.

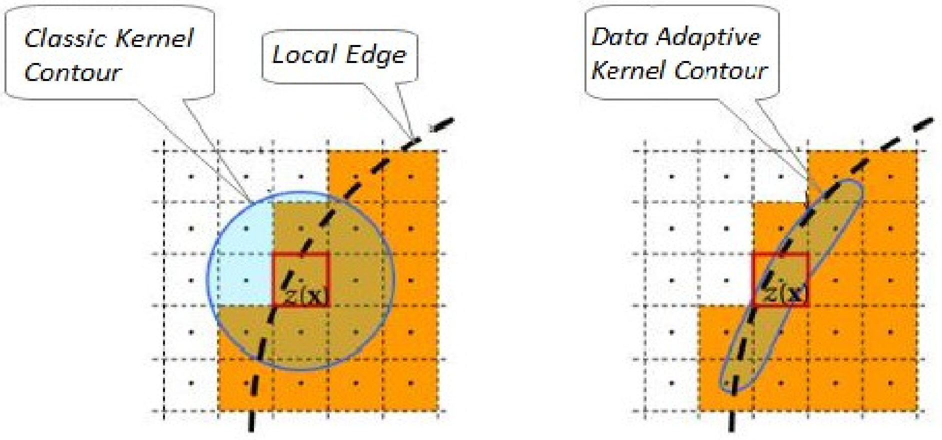
3.1 Data-Adaptive Kernels

Data-adaptive kernel regression relies on not only the spatial properties (the sample location and density), but also the photometric properties of these samples (i.e. pixel values). Thus, the effective size and shape of the regression kernel are adapted locally to image feature such as edges. A desired property of such regression kernel is illustrated in Figure 3.1, in which the classic kernel estimates the pixel z(x) by the combination of neighboring samples with linear weights while the data-adaptive kernel elongates/spreads along the local edge structure and the estimate is most strongly affected by the edge pixels. Hence, the data-adaptive kernel approach effectively suppresses noise while preserving local image structures. Data-adaptive kernel regression is formulated as an optimization problem where the data-adaptive kernel function

$$min_{\{B_n\}} \Sigma_{i=1}^{P} [y_i - B_0 - B_1^T (x_i - x) - B_2^T vech\{(x_i - x)(x_i - x)^T\} - \ldots]^2 K_{adapt}(x_i - x, y_i - y)$$

(3.1)

K_{adapt} now depends on the spatial sample co-ordinates x_i's and density as well as the photometric values y_i of the data. In the following, we study two different data-adaptive kernels: bilateral kernel and steering kernel, and discuss their properties.



(a) Contour of classic kernel **(b)** Contours of data-adaptive kernels

Figure 3.1: Kernel contours in a uniformly sampled data set: (a) Kernels in the classic method depend only on the spatial distances, and (b) Data-adaptive kernels elongate with respect to the local edge structure.

3.1.1 Bilateral Kernel

A simple and intuitive choice of the K_{adapt} is to use separate terms for penalizing the spatial distance between the pixel position of interest x and its neighboring pixel positions $\{x_i\}$, and the photometric "distance" between the pixel of interest y and its neighbors $\{ y_i \}$:

$$K_{bilat}(x_i - x, y_i - y) = K_{H_s}(x_i - x) . K_{h_p}(y_i - y) .$$

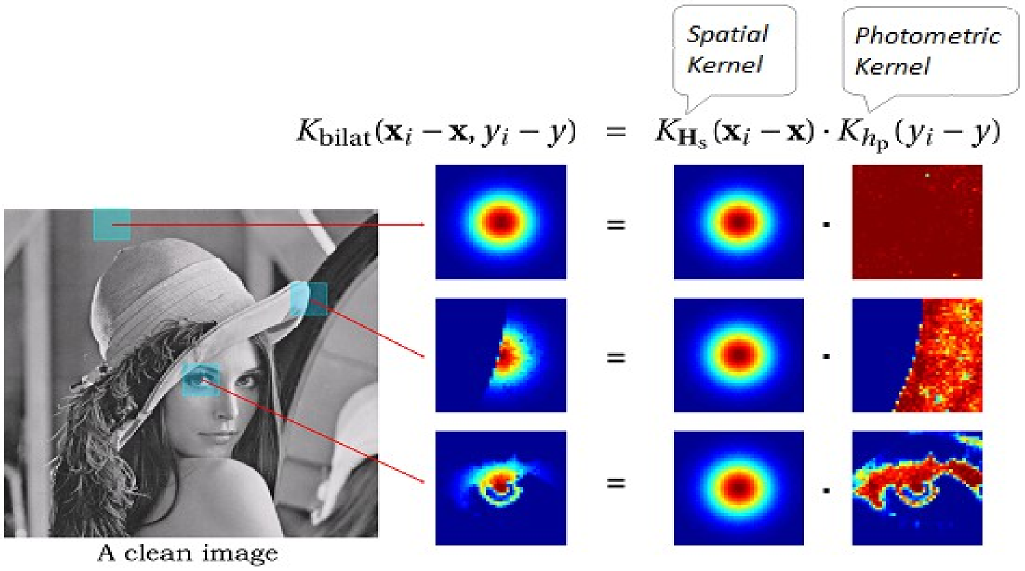
where H_s (= $h_s\mathbf{I}$) is the spatial smoothing (diagonal) matrix and h_p is the photometric smoothing scalar. Figure 3.2 illustrates weight values for this bilateral kernel function at a few different regions of the clean Lena image: flat, edge, and Lena's eye. As seen in the figure, the photometric kernel captures local image structures effectively. The properties of this adaptive method, which we call bilateral kernel regression (for reasons that will become clear shortly), can be better understood by studying the special case of the zeroth order ($N = 0$), which results in a data-adapted version of the Nadaraya-Watson estimator (NWE):

$$\hat{z}(x) = \hat{B}_0 = \frac{\Sigma_{i=1}^{P} K_{H_s}(x_i - x) K_{h_p}(y_i - y) y_i}{\Sigma_{i=1}^{P} K_{H_s}(x_i - x) K_{h_p}(y_i - y)} \tag{3.3}$$

Interestingly, this is nothing but the well-studied and popular bilateral filter. We note that, in general, since the pixel values (y) at an arbitrary position (x) might be unavailable from the given data, the direct application of the bilateral kernel function (3.2) is limited to the denoising problem.

This limitation, however, can be overcome by using an initial estimate of y by an appropriate interpolation technique. Also, it is worth noting that the bilateral kernel choice, along with higher order choices for N (> 0), will lead to generalizations of the bilateral filter.

Similar to classic kernel regression, the pixel estimator given by bilateral kernel regression is also summarized as the form of the weighted linear combination of all the neighboring samples using the bilateral "equivalent" weight function W_i regardless of the regression order N as follows:


Spatial Kernel
Photometric Kernel
$K_{\text{bilat}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{\mathbf{H}_s}(\mathbf{x}_i - \mathbf{x}) \cdot K_{h_p}(y_i - y)$
A clean image
Figure 3.2: Bilateral kernel weights given by (2.2) at flat, edge, and Lena's eye regions of a clean image.

$$\hat{z}(x) = \hat{\mathrm{B}}_0 = \Sigma_{i=1}^{P} W_i(K, H_S, h_p, N, x_i - x, y_i - y)\, y_i \tag{3.4}$$

Figure 3.3 illustrates the bilateral equivalent weight function W_i (K , H_s , h_p, N , $x_i - x$, $y_i - y$) in (3.4) at a variety of image structures for the zeroth and second orders ($N = 0$ and 2). Note that each weigh function is respectively normalized. Figure 3.4 illustrates the details of W_i at the strong edge:

As a further extension of the standard bilateral filter, iterative filtering in order to intensify the smoothing effect. The iterative filtering process is as follows: (i) apply bilateral filter to the given noisy data, (ii) apply bilateral filter to the previous estimate, (iii) repeat the step (ii). For $N = 0$, such estimator can be written as

$$\hat{z}^{(l+1)}(x)=\frac{\Sigma_{i=1}^{P} K_{H_s}(x_i-x)\, K_{h_p}(\hat{z}^{(l)}(x_i)-\hat{z}^{(l)}(x))\, \hat{z}_{(l)}(x_i)}{\Sigma_{i=1}^{P} K_{H_s}(x_i-x)\, K_{h_p}(\hat{z}^{(l)}(x_i)-\hat{z}^{(l)}(x))} \tag{3.5}$$

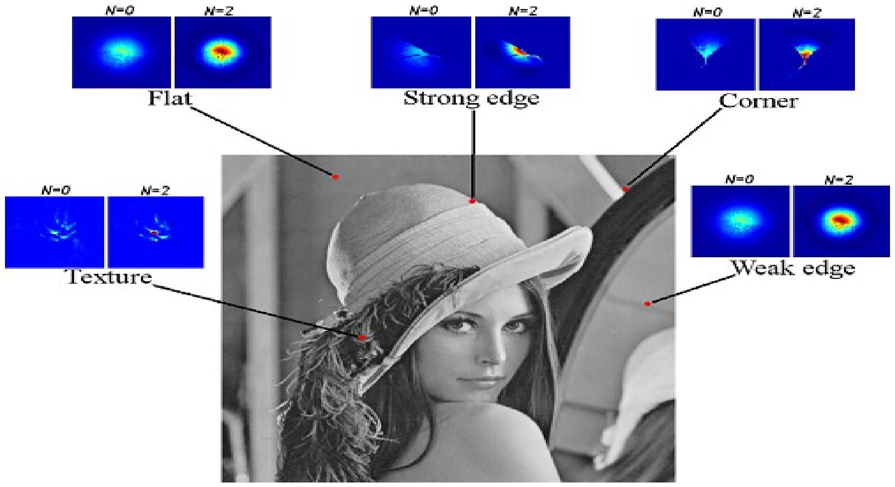


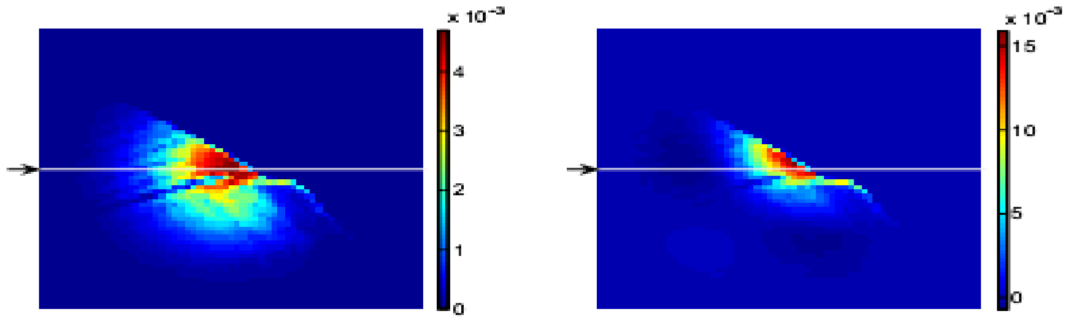
Figure 3.3: A visual analysis of the bilateral equivalent weight function Wi (K , Hs , hp, N , xi −x, yi−y) in (3.4) at a variety of image structures; flat, strong edge, corner, texture, and weak edge for the zeroth and second order (N = 0 and 2).

where $Z(0)(x_i) = y_i$ and ℓ is the index of the number of iterations. This filtering algorithm is very similar to Mean-Shift algorithm, in which the spatial kernel function $K_{hs}\ (x_i - x)$ is not taken into account.

The bilateral filter has appeared in another form (3.3), which is known as the Susan filter. The difference between bilateral filter and Susan filter is minor; Susan filter excludes the center pixel from the estimates. That is to say, Susan filter is expressed as

$$\hat{z}(x)=\hat{\mathrm{B}}_0=\frac{\Sigma_{x_i\neq x} K_{H_s}(x_i-x) K_{H_p}(y_i-y)\, y_i}{\Sigma_{x_i\neq x} K_{H_s}(x_i-x) K_{H_p}(y_i-y)} \tag{3.6}$$

This small modification significantly improves the filter performance in particular when the given data contains a few outliers (e.g. salt & pepper noise). For the bilateral filter, such



(a) Zeroth order **(b)** Second order

Figure 3.4: Horizontal cross-sections of the bilateral equivalent weight function W_i (K, $\mathbf{H}_s$, h_p, N , $x_i - x$, $y_i - y$) at the strong edge for the zeroth and second order (N = 0and 2): (a)-(b) the footprints of W_i for the zeroth and second order, respectively.

outlier pixels yields very small photometric kernel values for neighboring pixels because the photometric distances, $y_i - y$, tend to be large. In other words, the bilateral filter doesn't smooth an outlier pixel with its neighboring pixels.

In any event, breaking K adapt into spatial and photometric terms as utilized in the bilateral case weakens the estimator performance since it limits the degrees of freedom and ignores correlations between positions of the pixels and their values. In particular, we note that, for very noisy data sets, the photometric distances, $y_i - y$, tend to be large and noisier. Therefore, most photometric weights are close to zero and also noisy as shown in Figure 3.5. Such weights are effectively useless. Although we could set the photometric smoothing parameter (h_p) larger in order to reduce the effect of the noisy photometric distances, the bilateral filter becomes almost equivalent to the non-linear (Gaussian low-ass) filter with a large h_p. The following section provides a general solution to overcome this and many other drawbacks of competing approaches.

3.1.2 Steering Kernel

The filtering procedure that we propose next takes the data-adaptive idea one step further, based upon the earlier nonparametric framework. In particular, we observe that the effect of computing the photometric kernel, $K_{hp}\ (y_i - y)$ in (3.2) is to implicitly measure a function of the local

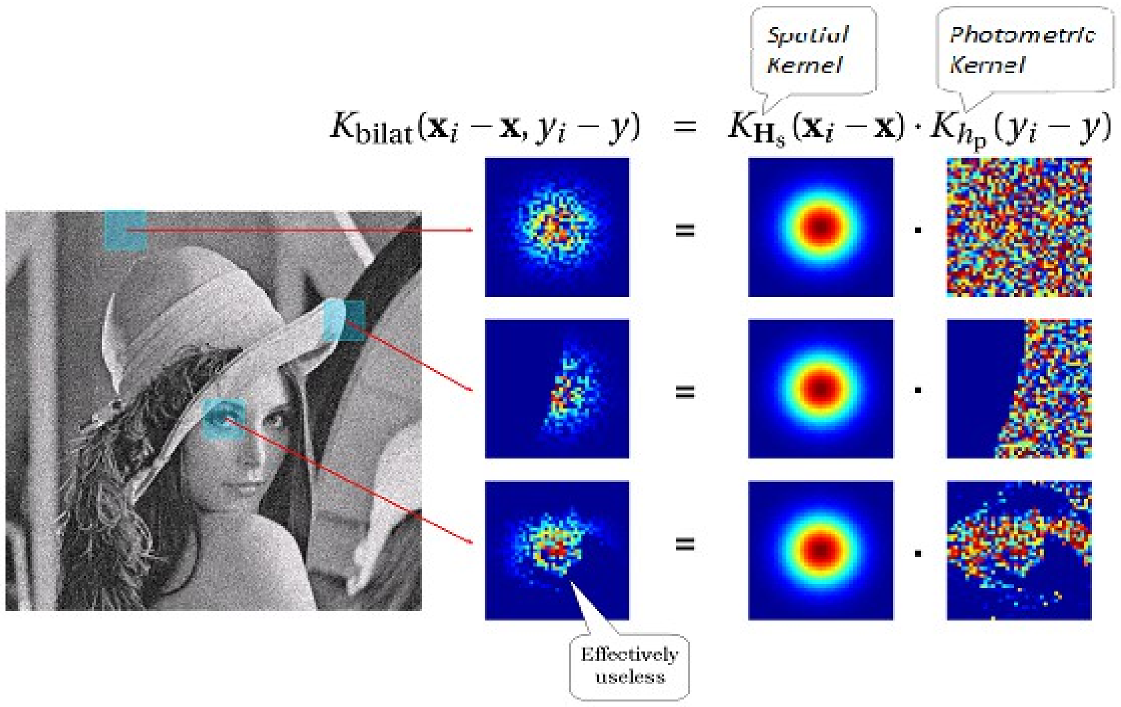

Spatial Kernel
Photometric Kernel
$K_{\mathrm{bilat}}(\mathbf{x}_i - \mathbf{x}, y_i - y) = K_{\mathbf{H}_s}(\mathbf{x}_i - \mathbf{x}) \cdot K_{h_p}(y_i - y)$
Effectively useless

Figure 3.5: Bilateral kernel weights given by (3.2) at flat, edge, and Lena's eye regions of a noisy image. The noisy image is given by adding white Gaussian noise.

gradient estimated between neighboring pixels and to use this estimate to weight the respective measurements. As an example, if a pixel is located near an edge, then pixels on the same side of the edge will have much stronger influence in the filtering. With this an initial estimate of the image gradients is made using some kind of gradient estimator (say the second order classic kernel regression method). Next, this estimate is used to measure the dominant orientation of the local gradients in the image. In a second filtering stage, this orientation information is then used to adaptively "steer" the local kernel, resulting in elongated, contours spread along the directions of the local edge structure. With these locally adapted kernels, the denoising is effected most strongly along the edges, rather than across them, resulting in strong preservation of details in the final output. To be more specific, the data-adaptive kernel function takes the form

$$K_{steer}(\boldsymbol{x_i}-\boldsymbol{x}, y_i-y)=K_{H_i}^{steer}(x_i-x) \tag{3.7}$$

where H_i^{steer}'s are now the data-dependent full (2 × 2) matrices which we call steering matrices. We define them as

$$H_i^{steer}=h\mu_i C_i^{\frac{-1}{2}} \tag{3.8}$$

where again *h* and μ_i are the global smoothing parameter and the local density parameter, respectively, and C_i's are (symmetric, 2 × 2) covariance matrices based on differences in the local gray-values. A good choice for C_i will effectively spread the kernel function along the local edges, as shown in Figure 3.1(b). It is worth noting that, even if we choose a large *h* in order to have a strong denoising effect, the undesirable blurring effect, which would otherwise have resulted, is tempered around edges with appropriate choice of C_i. With such steering matrices, for example, if we choose a Gaussian kernel, i.e. plugging the steering matrix (3.8) into Gaussian kernel function, the steering kernel is mathematically represented as

$$K_{H_i^{steer}}(\boldsymbol{x_i}-\boldsymbol{x})=\frac{\sqrt{det(C_i)}}{2\pi h^2\mu^2}\exp\{\frac{-(x_i-x)^T C_i(x_i-x)}{2\pi h^2\mu^2}\} \tag{3.9}$$

It is also noteworthy that, for the estimate of the unknown pixel $\beta_0(= z\,(x))$, the steering kernel function takes all the steering matrices (H_i^{steer}) of the neighboring pixels y_i around the position of interest x into account, and hence, the steering kernel is not simply elliptic but it provides us weights that fit the local image structures more flexibly. We will show some actual steering kernels shortly in this section.

The local edge structure is related to the gradient covariance (or equivalently, the locally dominant orientation), where a naive estimate of this covariance matrix may be obtained as follows:

$$C_i^{naive}=J_i^T J_i \tag{3.10}$$

where Ji is a stack of local gradient vectors in a local analysis window ω_i:

$$J_i = \begin{bmatrix} \vdots & \vdots \\ z_{x_1}(x_j) & z_{x_2}(x_j) \\ \vdots & \vdots \end{bmatrix} \tag{3.11}$$

$z_{x1}(\cdot)$ and $z_{x2}(\cdot)$ are the first derivatives along x_1 (vertical) and x_2 (horizontal) directions, and ω_i is a local analysis window around the position of a given sample. The dominant local orientation of the gradients is then related to the eigenvectors of this estimated matrix. Since the gradients, $z_{x1}(\cdot)$ and $z_{x2}(\cdot)$, depend on the pixel values $\{y_i\}$, and since the choice of the localized kernels in turns depends on these gradients, it, therefore, follows that the "equivalent" kernels for the proposed data-adaptive methods form a locally "nonlinear" combination of the data:

$$\hat{z}(x) = \Sigma_{i=0}^{P} W_i(K, H_i^{steer}, N, x_i - x)\, y_i \tag{3.12}$$

While the above approach to computing the steering matrices, is simple and has nice tolerance to noise, the resulting estimate can be unstable, and, therefore, care must be taken not to take the inverse of the estimate directly in this case. In such case, a diagonal loading or regularization methods can be used to obtain stable estimates of the covariance. We take a more robust approach to the design of the steering matrix.

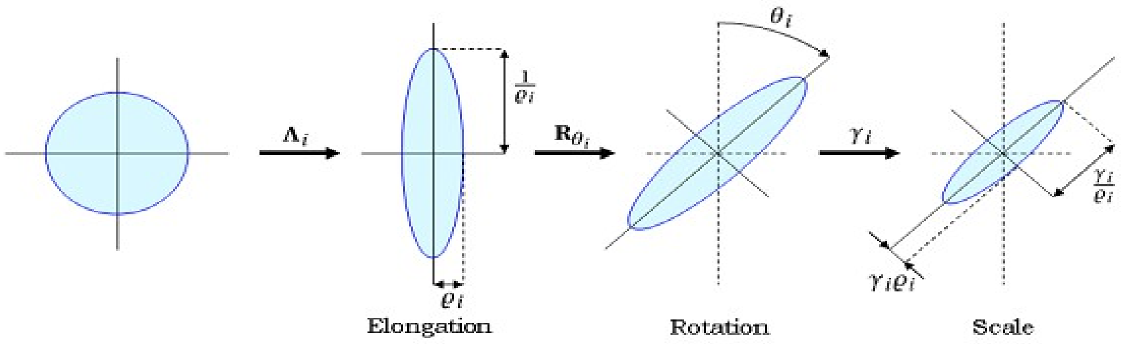


Figure 3.6: A schematic representation illustrating the effects of the steering matrix and its components $C_i = \gamma_i R_{\theta_i} \Lambda_i R_{\theta_i}^T$ on the size and shape of the regression kernel footprint.

In order to have a more convenient form of the covariance matrix, we decompose it into three components (equivalent to eigenvalue decomposition) as follows:

$$C_i = \gamma_i R_{\theta_i} \Lambda_i R_{\theta_i}^T \tag{3.13}$$

where $R_{\theta i}$ is the rotation matrix and Λ_i is the elongation matrix:

$$R_{\theta_i} = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix} , \qquad \Lambda_i = \begin{bmatrix} \varrho_i & 0 \\ 0 & \frac{1}{\varrho_i} \end{bmatrix} \tag{3.14}$$

Now, the covariance matrix given by the three parameters γ_i, θ_i , and ϱ_i, which are the scaling, rotation, and elongation parameters, respectively. Figure 3.6 schematically explains how these parameters affect the spreading of kernels. First, the circular kernel is elongated by the elongation matrix Λ_i , and its semi-minor and major axes are given by ϱ_i. Second, the elongated kernel is rotated by the matrix $R_{\theta i}$. Finally, the kernel is scaled by the scaling parameter γ_i.

We define the scaling, elongation, and rotation parameters as follow. The dominant orientation of the local gradient field is the singular vector corresponding to the smallest (nonzero) singular value of the local gradient matrix J_i (3.11) arranged in the following form:

$$J_i = U_i S_i V_i^T = U_i \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} v_1 & v_2 \end{bmatrix}^T \tag{3.15}$$

where $\mathbf{U}_i\mathbf{S}_i\mathbf{V}_i^T$ is the truncated singular value decomposition of J_i , and $\boldsymbol{S}_i$ is a diagonal 2 × 2 matrix representing the energy in the dominant directions. Then, the second column of the 2 × 2 orthogonal matrix $\mathbf{V}_i$, $v_2 = [v_{12}, v_{22}]^T$, defines the dominant orientation angle θ_i as

$$\theta_i = \arctan\left(\frac{v_{12}}{v_{22}}\right) \tag{3.16}$$

That is, the singular vector corresponding to the smallest nonzero singular value s_2 of J_i represents the dominant orientation of the local gradient field. The elongation parameter ϱ_i can be scaled corresponding to the energy of the dominant gradient direction

$$\varrho_i = \frac{s_1 + \lambda'}{s_2 + \lambda'}, \lambda' \geq 0, \tag{3.17}$$

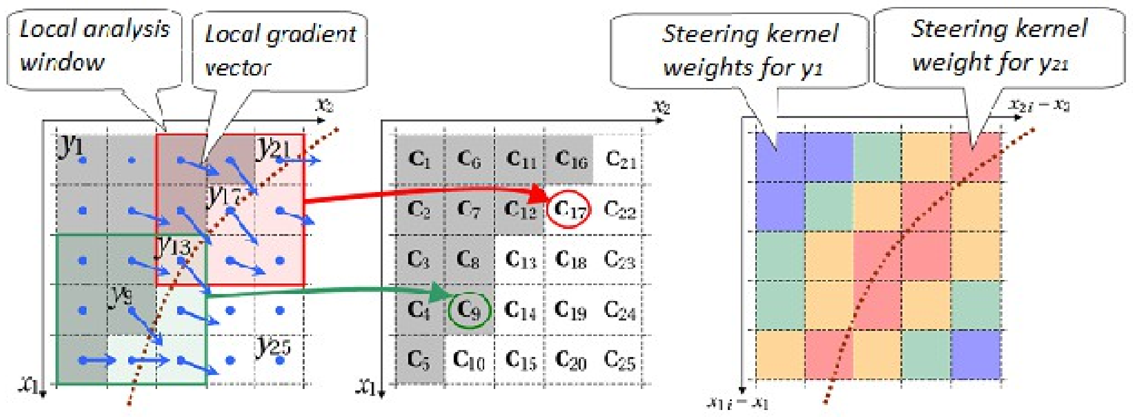
where λ' is a “regularization” parameter for the kernel elongation, which dampens the effect of the noise and restricts the ratio from becoming degenerate. The intuition behind (3.17) is to keep the shape of the kernel circular in flat areas ($s_1 \approx s_2 \approx 0$), and elongate it near edge areas ($s_1 >> s_2$). Finally, the scaling parameter γ_i is defined by

$$\gamma_i = \left(\frac{s_1 S_2 + \lambda''}{M} \right)^{\alpha} \tag{3.18}$$

where λ'' is again a "regularization" parameter, which dampens the effect of the noise and keeps γ_i from becoming zero, M is the number of samples in the local analysis window, and α is the structure sensitivity parameter. The intuition behind (3.18) is that, to reduce noise effects while producing sharp images, large footprints are preferred in the flat (smooth regions) and smaller ones in the textured areas. Note that the local gradients and the eigenvalues of the local gradient matrix $C_i^{naïve}$ are smaller in the flat (low-frequency) areas than the textured (high-frequency) areas. As the product s_1 s_2 is the geometric mean of the eigenvalues of $C_i^{naïve}$ γ_i makes the steering kernel area large in the flat, and small in the textured areas. The structure sensitivity α (typically $0 \leq \alpha \leq 0.5$) controls how strongly the size of the kernel footprints is affected by the local structure. The product of the singular values indicates the amount of energy of the local signal structure: the larger the product, the stronger and the more complex the local structure is. A large α is preferable when the given signal carries severe noise.

Figure 3.7 illustrates a schematic representation of the estimate of local co-variance matrices and the computation of steering kernel weights for the center pixel y_{13}. First, we estimate the gradients and compute the local covariance matrix C_i by (3.13)-(3.18) for each pixel.

We compute the steering kernel weights for each neighboring pixel with its C_i. In this case, even though the spatial distances from y_{13} to y_1 and to y_{21} are equal, the steering kernel weight for y_{21} (i.e. $K_{H21}(x_{21} - x_{13})$) is larger than the one for y_i (i.e. $K_{H1}(x_1 - x_{13})$). Moreover, as Figure 3.8(a) illustrates the steering kernel weights on a variety of image structures of a clean Lena image, weights given by the steering kernel function (3.9) with (3.13) captures local structures more effectively. This is because the steering kernel function is a function of the position of neighboring samples (x_i) with the position of interest (x) held fixed. Each neighboring sample (y_i) has a steering matrix (H_i^{steer}), and,



(a) A covariance matrix from local gradients with 3 × 3 analysis window **(b)**Steering kernel weights

Figure 3.7: A schematic representation of the estimates of local covariance metrics and the steering kernel weights at a local region with one dominant orientation: (a) First, we estimate the gradients and compute the local covariance matrix C_i by (2.13) for each pixel, and (b) Next, for y_{13}, we compute the steering kernel weights with C_i for neighboring pixels. Even though, in this case, the spatial distances between y_{13} and y_1 and between y_{13} y_{21} are equal, the steering kernel weight for y_{21} (i.e. $K_{H21}(x_{21} - x_{13})$) is larger than the one for y_1 (i.e. $K_{H1}(x_1 - x_{13})$). This is because y_{13} and y_{21} are located along the same edge.

unlike the adaptive normalized convolution method, the steering kernel function takes not only the steering matrix at the position of interest but also its neighborhoods' into account. As a result, the steering kernel has more flexibility to adapt to local image structures. This property is effective for object recognition applications.

Chapter 4

Principal component analysis

In many problems, the measured data vectors are high-dimensional but we may have reason to believe that the data lie near a lower-dimensional manifold. In other words, we may believe that high-dimensional data are multiple, indirect measurements of an underlying source, which typically cannot be directly measured. Learning a suitable low-dimensional manifold from high-dimensional data is essentially the same as learning this underlying source.

Dimensionality reduction can also be seen as the process of deriving a set of degrees of freedom which can be used to reproduce most of the variability of a data set. Consider a set of images produced by the rotation of a face through different angles. Clearly only one degree of freedom is being altered, and thus the images lie along a continuous one-dimensional curve through image space.

Manifold learning techniques can be used in different ways including:

1. Data dimensionality reduction: Produce a compact low-dimensional encoding of a given high dimensional data set.
2. Data visualization: Provide an interpretation of a given data set in terms of intrinsic degree of freedom, usually as a by-product of data dimensionality reduction.
3. Pre-processing for supervised learning: Simplify, reduce, and clean the data for subsequent supervised training.

Many algorithms for dimensionality reduction have been developed to accomplish these tasks. However, since the need for such analysis arises in many areas of study, contributions to the field have come from many disciplines. While all of these methods have a similar goal, approaches to the problem are different.

Principal components analysis (PCA) [12] is a classical method that provides a sequence of best linear approximations to a given high-dimensional observation. It is one of the most popular techniques for dimensionality reduction. However, its effectiveness is limited by its global linearity. Multidimensional scaling (MDS) [13], which is closely related to PCA, suffers from the same drawback. Independent component analysis (ICA) [14] also assume that the underling manifold is a linear subspace. However, they differ from PCA in the way they identify and model the subspace. The subspace modeled by PCA captures the maximum variability in the data, and can be viewed as modeling the covariance structure of the data, whereas factor analysis models the correlation structure. ICA starts from a factor analysis solution and searches for rotations that lead to independent components. The main drawback with all these classical dimensionality reduction approaches is that they only characterize linear subspaces (manifolds) in the data.

4.1 Principal Components Analysis

Principal components analysis (PCA) is a very popular technique for dimensionality reduction. Given a set of data on n dimensions, PCA aims to find a linear subspace of dimension d lower than n such that the data points lie mainly on this linear subspace (See Figure 1.2 as an example of a two-dimensional projection found by PCA). Such a reduced subspace attempts to maintain most of the variability of the data.

The linear subspace can be specified by *d* orthogonal vectors that form a new coordinate system, called the `principal components'. The principal components are orthogonal, linear transformations of the original data points, so there can be no more than n of them. However, the hope is that only d < n principal components are needed to approximate the space spanned by the n original axes. The most common definition of PCA, due to Hotelling [15], is that, for a given set of data vectors $x_i, i \in 1 \ldots t$, the *d* principal axes are those orthonormal axes onto which the variance retained under projection is maximal. In order to capture as much of the variability as possible, let us choose the first principal component, denoted by U_1, to have maximum variance. Suppose that all centered observations are stacked into the columns of an $n \times t$ matrix *X,* where each column corresponds to an n-dimensional observation and there are t observations. Let the first principal component be a linear combination of *X* defined by coefficients (or weights) $w = [w_1 \ldots w_n]$. In matrix form:

$$U_1 = w^T X$$

$$var(U_1) = var(w^T X) = w^T S w$$

where *S* is the n × n sample covariance matrix of *X*.

Clearly var(U_1) can be made arbitrarily large by increasing the magnitude of *w.*

Therefore, we choose w to maximize $w^T Sw$ while constraining w to have unit length.

$$\max \quad w^T Sw$$
$$\text{subject to} \quad w^T w = 1$$

To solve this optimization problem a Lagrange multiplier α_1 is introduced:

$$L(w, \alpha) = w^T Sw - \alpha_1 (w^T x - 1) \tag{4.1}$$

Differentiating with respect to w gives n equations,

$$Sw = \alpha_1 w$$

Pre-multiplying both sides by w^T we have:

$$w^T Sw = \alpha_1 w^T w = \alpha_1$$

var(U_1) is maximized if α_1 is the largest eigenvalue of S.

Clearly α_1 and w are an eigenvalue and an eigenvector of S. Differentiating (4.1) with respect to the Lagrange multiplier α_1 gives us back the constraint:

$$w^T w = 1$$

This shows that the first principal component is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix S. A similar argument can show that the d dominant eigenvectors of covariance matrix S determine the first d principal components.

Another nice property of PCA, is that the projection onto the principal subspace minimizes the squared reconstruction error, $\sum_{i=1}^{t} \|x_i - \hat{x}_i\|^2$ .In other words, the principal components of a set of data in $\mathbb{R}^n$ provide a sequence of best linear approximations to that data, for all ranks $d \leq n$

Consider the rank-d linear approximation model as:

$$f(y) = \bar{x} + U_d y$$

This is the parametric representation of a hyperplane of rank d.

For convenience, suppose $x = 0$ (otherwise the observations can be simply replaced by their centered versions $\tilde{x} = x_i - \bar{x}$). Under this assumption the rank d linear model would be $f(y) = U_d y$, where U_d is a $n \times d$ matrix with d orthogonal unit vectors as columns and y is a vector of parameters. Fitting this model to the data by least squares leaves us to minimize the reconstruction error:

$$\min_{U_d, y_i} \sum_{i}^{t} \left\| x_i - U_d y_i \right\|^2$$

By partial optimization for y_i we obtain:

$$\frac{d}{dy_i} = 0 \Rightarrow y_i = U_d^T x_i$$

Now we need to find the orthogonal matrix U_d:

$$\min_{U_d} \sum_{i}^{t} \left\| x_i - U_d D_d^T x_i \right\|^2$$

Define $H_d = U_d U_d^T$. H_d is an n× n matrix which acts as a projection matrix and projects each data point xi onto its rank d reconstruction. In other words, H_d x_i is the orthogonal projection of x_i onto the subspace spanned by the columns of U_d. A unique solution U can be obtained by finding the singular value decomposition of X [17]. For each rank d, U_d consists of the first d columns of U.

Clearly the solution for U can be expressed as singular value decomposition (SVD) of X.

$$X = U \Sigma V^T$$

since the columns of U in the SVD contain the eigenvectors of XX^T. The PCA procedure is summarized in Algorithm 4.1

Algorithm 4.1

Recover basis: Calculate $XX^T = \sum_{i=1}^{t} x_i x_i^T$ and let U = eigenvectors of XX^T corresponding to the top d eigenvalues.

Encode training data: $Y = U^T X$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = UU^T X$

Encode test example: $y = U^T x$ where y is a d-dimensional encoding of x.

Reconstruct test example: $\hat{x} = Uy = UU^T x$

Table 4.1: Direct PCA Algorithm

4.1.1 Dual PCA

It turns out that the singular value decomposition also allows us to formulate the principle components algorithm entirely in terms of dot products between data points and limit the direct dependence on the original dimensionality *n*. This fact will become important below.

Assume that the dimensionality *n* of the $n \times t$ matrix of data *X* is large (i.e., n >> t). In this case, Algorithm 4.1 (Table 4.1) is impractical. We would prefer a run time that depends only on the number of training examples *t*, or that at least has a reduced dependence on n.

Note that in the SVD factorization $X = U\Sigma V^T$, the eigenvectors in *U* corresponding to nonzero singular values in Σ (square roots of eigenvalues) are in a one-to-one correspon-dence with the eigenvectors in *V*. Now assume that we perform dimensionality reduction on *U* and keep only the first *d* eigenvectors, corresponding to the top *d* nonzero singular values in Σ. These eigenvectors will still be in a one-to-one correspondence with the first *d* eigenvectors in *V*:

$$XV = U\Sigma$$

where the dimensions of these matrices are:

$$\begin{array}{cccc} X & U & \Sigma & V \\ n\times t & n\times d & d\times d & t\times d \\ & & \textit{diagonal} & \end{array}$$

Crucially, Σ is now square and invertible, because its diagonal has nonzero entries. Thus, the following conversion between the top *d* eigenvectors can be derived:

$$U = XV\Sigma^{-1} \qquad (4.2)$$

Replacing all uses of *U* in Algorithm 1 with $XV\Sigma^{-1}$ gives us the dual form of PCA, Algorithm 2 (see Table 4.2). Note that in Algorithm 4.2 (Table 4.2), the steps of "Reconstruct training data" and "Reconstruction test example" still depend on *n*, and therefore still will be impractical in the case that the original dimensionality *n* is very large. However all other steps can be done conveniently in the run time that depends only on the number of training examples *t*.

Algorithm 4.2

U eigenvectors, *X*

Recover basis: Calculate X^TX and let V = eigenvectors of X^TX corresponding to the top d eigenvalues. Let Σ = diagonal matrix of square roots of the top d eigenvalues.

Encode training data: $Y = U^TX = \Sigma\ V^T$ where Y is a $d \times t$ matrix of encodings of the original data.

Reconstruct training data: $\hat{X} = UY = U\Sigma V^T = XV\Sigma^{-1}\Sigma V^T = XVV^T$

Encode test example: $y = U^T x = \Sigma^{-1} V^T X^T x = XV^T X^T x$ where y is a d dimensional encoding of x.

Reconstruct test example: $\hat{x} = Uy = UU^T x = XV\Sigma^{-2} V^T X^T x = XV\Sigma^{-2} V^T X^T x$

Table 4.2: Dual PCA Algorithm

4.2 Kernel PCA

PCA is designed to model linear variabilities in high-dimensional data. However, many high dimensional data sets have a nonlinear nature. In these cases the high-dimensional data lie on or near a nonlinear manifold (not a linear subspace) and therefore PCA can not model the variability of the data correctly. One of the algorithms designed to address the problem of nonlinear dimensionality reduction is Kernel PCA. In Kernel PCA, through the use of kernels, principle components can be computed efficiently in high-dimensional feature spaces that are related to input space by some nonlinear mapping.

Kernel PCA finds principal components which are nonlinearly related to the input space by performing PCA in the space produced by the nonlinear mapping, where the low-dimensional latent structure is, hopefully, easier to discover.

Consider a feature space $\mathcal{H}$ such that:

$$\Phi : x \rightarrow \mathcal{H}$$

$$x \rightarrow \Phi(x)$$

Suppose $\Sigma_i^t \Phi(x_i) = 0$. This allows us to formulate the kernel PCA objective as follows:

$$min \sum_{i}^{t} \left\| \Phi(x_i) - U_q U_q^T \Phi(x_i) \right\| \tag{4.3}$$

By the same argument used for PCA, the solution can be found by SVD:

$$\Phi(X) = U \Sigma V^T \tag{4.4}$$

where U contains the eigenvectors of $\Phi(X)\Phi(X)^T$. Note that if Φ(X) is $n \times t$ and the dimensionality of the feature space n is large, then U is $n \times n$ which will make PCA impractical.

To reduce the dependence on n, first assume that we have a kernel $K(.,.)$ that allows us to compute $K(x,y) = \Phi(x)^T \Phi(y)$ Given such a function, we can then compute the matrix $\Phi(X)^T \Phi(X) = K$ efficiently, without computing Φ(X) explicitly. Crucially, K is $t \times t$ here and does not depend on n. Therefore it can be computed in a run time that depends only on t. Also, note that PCA can be formulated entirely in terms of dot products between data points (Algorithm 2 represented in Table 1.2). Replacing dot products in Algorithm 2 (1.2) by kernel function K, which is in fact equivalent to the inner product of a Hilbert space yields to the Kernel PCA algorithm.

4.2.1 Centering

In the derivation of the kernel PCA we assumed that Φ(X) has zero mean. The following normalization of the kernel satisfies this condition.

$$\tilde{K}(x,y) = K(x,y) - E_x[K(x,y)] - E_y[K(x,y)] + E_X[E_y[K(x,y)]] \tag{4.5}$$

In order to prove that, define:

$$\tilde{\Phi}(X) = \Phi(x) - E_x[\Phi(X)] \tag{4.6}$$

Finally, the corresponding kernel is:

$$\tilde{K}(x,y) = \tilde{\Phi}(x)\tilde{\Phi}(y) \tag{4.7}$$

This expands as follows:

$$\begin{aligned}\tilde{K}(x,y) &= (\Phi(x) - E_x[\Phi(x)]) \cdot (\Phi(y) - E_y[\Phi(y)]) \\ &= K(x,y) - E_x[K(x.y)] - E_y[K(x,y)] + E_x[E_y[K(x,y)]]\end{aligned} \tag{4.8}$$

To perform Kernel PCA, one needs to replace all dot products $x^T y$ by $\tilde{K}(x,y)$ in Algorithm 2 (Table 1.2). Note that V is the eigenvectors of *K(X;X)* corresponding to the top *d* eigenvalues, and Σ is diagonal matrix of square roots of the top *d* eigenvalues. Unfortunately Kernel PCA does not inherit all the strength of PCA. More specifically reconstruction of training and test data points is not a trivial practice in Kernel PCA. Algorithm 2 (Table 1.2) shows that data can be reconstructed in feature space Φ(x).

4.3 Proof of PCA

With the knowledge from linear algebra, we now can prove that PCA results in the best compression with the minimal loss of information. We use the following considerations in the proof:

The first consideration is:

$$E\left[\bar{x}^T \bar{x}\right]=E\left[\sum_{i=1}^{N} x_i^2\right]=\sum_{i=1}^{N} E\left[x_i^2\right]=\sum_{i=1}^{N} R_{ii} \qquad (I) \tag{4.9}$$

The second consideration, with

$$c_i = \bar{x}^T \bar{q}_i \tag{4.10}$$

$$\begin{aligned} E\left[c_i c_j\right] &= E\left[\bar{x}^T \bar{q}_i \bar{q}_j^T \bar{x}\right] \\ &= E\left[\bar{q}_j^T \bar{x}\bar{x}^T \bar{q}_i\right] \\ &= \bar{q}_j^T E\left[\bar{x}\bar{x}^T\right]\bar{q}_i \\ &= \bar{q}_j^T \bar{\bar{R}}\bar{q}_i \\ &= \bar{q}_j^T \lambda_i \bar{q}_i \\ &= \lambda_i \bar{q}_j^T \bar{q}_i \\ &= \lambda_i \delta_{ij} \end{aligned} \tag{4.11}$$

The difference between two random vectors with the same correlation matrix now is:

$$E\left[\|\bar{x}-\bar{\bar{x}}\|^2\right]=E\left[\left(\bar{x}-\sum_{i=1}^{N} c_i \bar{q}_i\right)^2\right]$$

$$= E\left[\left(\bar{x}-\sum_{i=1}^{N} c_i \bar{q}_i\right)^T\left(\bar{x}-\sum_{i=1}^{N} c_i \bar{q}_i\right)\right]$$

$$= E\left[\bar{x}^T \bar{x}\right]-2\sum_{i=1}^{N} E\left[c_i \bar{x}^T \bar{q}_i\right]+\sum_{j=1}^{N}\sum_{i=1}^{N} E\left[c_i c_j \bar{q}_i^T \bar{q}_j\right]\overrightarrow{(I)\wedge(II)}$$

$$= \sum_{i=1}^{N} R_{ii} - 2\sum_{i=1}^{N} E\left[c_i^2\right] + \sum_{j=1}^{N}\sum_{i=1}^{N} \lambda_i \delta_{ij}$$

$$= \sum_{i=1}^{N} R_{ii} - 2\sum_{i=1}^{N} \lambda_i + \sum_{j=1}^{N}\sum_{i=1}^{N} \lambda_i \delta_{ij}$$

$$= \sum_{i=1}^{N} R_{ii} - 2\sum_{i=1}^{N} \lambda_i + \sum_{i=1}^{N} \lambda_i$$

$$= \sum_{i=1}^{N} R_{ii} - \sum_{i=1}^{N} \lambda_i \qquad (4.12)$$

Since the matrix R is Hermitian this error equals 0, because for such a Hermitian matrix it holds that:

$$\sum_{i=1}^{N} R_{ii} = \sum_{i=1}^{N} \lambda_i \qquad (4.13)$$

Now we have proven that a measure for the difference or *error* between $\bar{x}$ and $\tilde{\tilde{x}}$ is given by:

$$E\left[\left\|\bar{x} - \tilde{\tilde{x}}\right\|^2\right] = \sum_{i=1}^{N} R_{ii} - \sum_{i=1}^{N} \lambda_i \qquad (4.14)$$

We may rewrite $\bar{x}$ as a combination of eigenvectors:

$$\bar{x} = \sum_{i=1}^{M} c_i \bar{q}_i + \sum_{i=M+1}^{N} c_i \bar{q}_i \qquad (4.15)$$

If we want to compress $\bar{x}$ the next question is, which terms can we best leave out?
The error by leaving out terms is expressed by:

$$\sum_{i=1}^{N} R_{ii} - \sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} R_{ii} - \sum_{i=1}^{M} \lambda_i - \sum_{i=M+1}^{N} \lambda_i \qquad (4.16)$$

Clearly, when we use the smallest eigenvalues λ_i for the part we leave out (the terms numbered M+1 until N), we obtain the smallest error.

Chapter 5

Resemblance Map and Significance Testing

An ensemble of local features with little discriminative power can together offer a significant discriminative power, both quantization and informative feature selection on a long-tail distribution can lead to a precipitous drop in performance. Therefore, instead of any quantization and informative feature selection, we focus on reducing the dimension of densely computed LSKs using PCA to enhance the discriminative power and reduce computational complexity. This idea results in a new feature representation with a moderate dimension which inherits the desirable discriminative attributes of LSK.

5.1 Feature representation:

In order to organize $W_Q^j(x_i - x)$ and $W_T^j(x_l - x)$, which are densely computed from *Q* and *T*, let W_Q, W_T be matrices whose columns are vectors w^j_Q, w^j_T, which are column-stacked (rasterized) versions of $w_Q^j(x_l - x), W_T^j(x_l - x)$ respectively:

$$W_Q = \left[w_Q^{1,} \cdots, w_Q^n\right] \in \mathbb{R}^{P^2 \times n}, \qquad W_T = \left[w_Q^{1,} \cdots, w_Q^{nT}\right] \in \mathbb{R}^{P^2 \times nT}$$

the next step is to apply PCA to W_Q for dimensionality reduction and to retain only its salient characteristics. Applying PCA to W_Q we can retain the first (largest) d principal components which form the columns of a matrix $A_Q \in \mathbb{R}^{P^2 \times d}$. Next, the lower dimensional features are computed by projecting W_Q and W_T onto A_Q

$$F_Q = \left[f_Q^{1,} \cdots, f_q^n\right] = A_Q^T W_Q \in \mathbb{R}^{d \times n}, \qquad F_Q = \left[f_Q^{1,} \cdots, f_q^{nT}\right] = A_Q^T W_Q \in \mathbb{R}^{d \times nT}$$

5.2 Matrix Cosine as a Measure of Similarity

The next step in the proposed framework is a decision rule based on the measurement of a "distance" between the computed features F_Q, F_{Ti}. Correlation based metrics outperforms the conventional Euclidean and Mahalanobis distances for the classification and subspace learning tasks. Motivated by the effectiveness of correlation-based similarity measure, we introduce “Matrix Cosine Similarity” for the matrix case and explore the idea behind this measure in this section. In general, “correlation” indicates the strength and direction of a linear relationship between two random variables. But the idea of correlation is quite malleable.

The Pearson’s correlation coefficient which is the familiar standard correlation coefficient, and the cosine similarity (so-called non-Pearson-compliant). Cosine similarity coincides with the Pearson’s correlation when each vector is centered to have zero-mean. It has been shown that the Pearson correlation is less discriminating than the cosine similarity due to the fact that centered values are less informative than the original values, and the computation of centered values is sensitive to zero or small values in the vectors. Since the discriminative power is critical in our detection framework, we focus on the cosine similarity. The cosine similarity is defined as the inner product between two normalized vectors as follows:

$$\rho(f_Q, F_{T_i}) = \langle \frac{f_Q}{\|f_Q\|}, \frac{f_{T_i}}{\|f_{T_i}\|} \rangle = \frac{F_Q^T f_{T_i}}{\|f_Q\| \ \|f_{T_i}\|} = \cos\theta_i, \in [-1, 1],$$

where $f_Q, f_{T_i} \in \mathbb{R}^d$ are column vectors. The cosine similarity measure therefore focuses only on the angle (phase) information while discarding the scale information.

The next step is to generate a so-called “resemblance map” (RM), which will be an image withvalues indicating the likelihood of similarity between the Q and T . When it comes to interpreting the value of “correlation”, it is noted that the proportion of variance in common between the two feature sets as opposed to ρ_i which indicates a linear relationship between two feature matrices F_Q, F_{Ti}. At this point, we can use ρ_i directly as a measure of resemblance between the two feature sets. However, the shared variance interpretation of ρ^2_i has several advantages. In particular, as for the final test statistic comprising the values in the resemblance map, we use the proportion of shared variance (ρ^2_i) to that of the “residual” variance ($1 - \rho^2_i$). More specifically, RM is computed using the mapping function f as follows:

$$\text{RM:} \quad f(\rho_i) = \frac{\rho_i^2}{1-\rho_i^2}$$

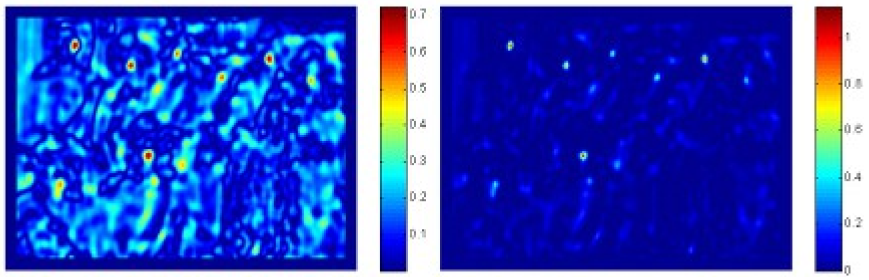

(a) (b)

Figure 5.1 (a) Resemblance map (RM) which consists of |ρi| (b) Resemblance map (RM) which consists of f (ρi).

In Figure 5.1, examples of resemblance map (RM) based on $|\rho_i|$ and $f(\rho_i)$ are presented. Red color represents higher resemblance. As is apparent from these typical results, qualitatively, the resemblance map generated from $f(\rho_i)$ provides better contrast and dynamic range in the result (f (ρi) ∈ [0, ∞]). More importantly, from a quantitative point of view, we note that $f(\rho_i)$ is essentially the Lawley-Hotelling Trace statistic, which is used as an efficient test statistic for detecting correlation between two data sets. Furthermore, it is worth noting that historically, this statistic has been suggested in the pattern recognition literature as an effective means of measuring the separability of two data clusters

5.3 Non-Parametric Significance Test and Non-Maxima Suppression

If the task is to find the most similar patch (*Ti*) to the query (*Q*) in the target image, one can choose the patch which results in the largest value in the RM (i.e., max $f(\rho_i)$) among all the patches, no matter how large or small the value is in the range of [0, ∞]. This, however, is not wise because there may not be any object of interest present in the target image. We are therefore interested in two types of significance tests. The first is an overall test to decide whether there is any sufficiently similar object present in the target image at all. If the answer is yes, we would then want to know how many objects of interest are present in the target image and where they are. Therefore, we need two thresholds: an overall threshold τ_o and a threshold τ to detect the possibly multiple objects present in the target image.

In a typical scenario, we set the overall threshold τo to be, for instance, 0.96 which is about 50% of variance in common (i.e., ρ_2 = 0.49). In other words, if the maximal $f(\rho_i)$ is just above 0.96, we decide that there exists at least one object of interest. The next step is to choose τ based on the properties of $f(\rho_i)$. When it comes to choosing the τ , there is need to be more careful. If we have a basic knowledge of the underlying distribution of $f(\rho_i)$, then we can make predictions about how this particular statistic will behave, and thus it is relatively easy to choose a threshold which will indicate whether the pair of features from the two images are sufficiently similar. But, in practice, we do not have a very good way to model the distribution of $f(\rho_i)$. Therefore, instead of assuming a type of underlying distribution, we employ the idea of nonparametric testing. We compute an empirical PDF from all the give samples of $f(\rho_i)$ and we set τ so as to achieve, for instance, a 99 % confidence level in deciding whether the given

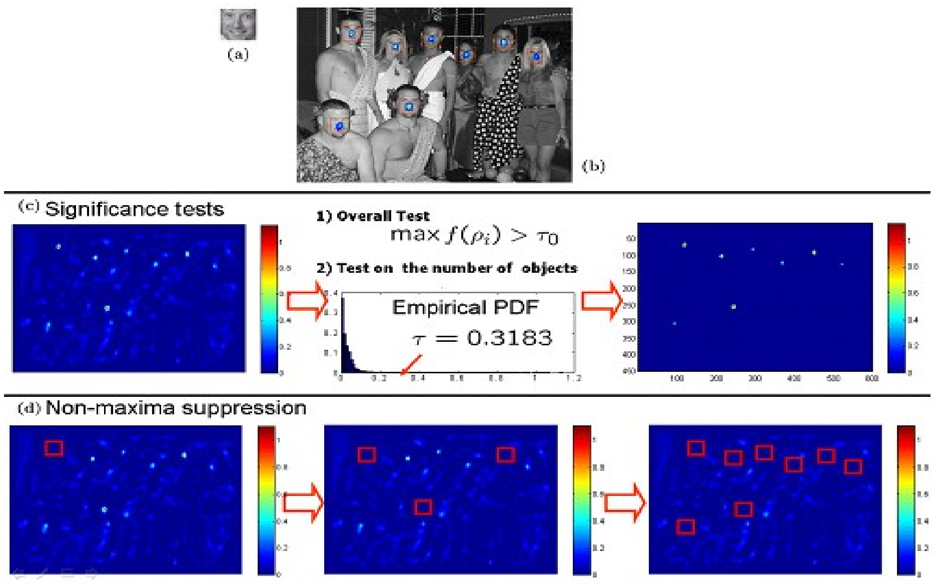

(a)
(b)
(c) Significance tests
1) Overall Test
$\max f(\rho_i) > \tau_0$
2) Test on the number of objects
Empirical PDF
$\tau = 0.3183$
(d) Non-maxima suppression

Figure 5.2 (a) Query (b) Target with detection (c) Two significance tests (d) Non-maxima suppression

values are in the extreme (right) tails of the distribution. This approach is based on the assumption that in the target image, most of patches do not contain the object of interest, and therefore, the few matches will result in values which are in the tails of the distributions of $f(\rho_i)$. After the two significance tests with τ_o, τ are performed, we employ the idea of non-maxima suppression for the final detection. We take the region with the highest $f(\rho_i)$ value and eliminate the possibility that any other object is detected within some radius of the center of that region again. This enables us to avoid multiple false detections of nearby objects already detected. Then we iterate this process until the local maximum value falls below the threshold τ. Fig. 5. shows the graphical illustration of significance tests and the non-maxima suppression idea.

Chapter 6

Methodology

For video indexing it is necessary that we first segment the video in frames for which the LARK algorithm could be used to work upon. Here we assume a new object appears in a new frame which causes an abrupt change in pixel values which is usually the case with a random videos. An abrupt change in video can be detected using histogram intersection.

Here we take the example of a Tank that we will try to identify in the video and will be used as a query here in the process. Our methodology involves the following steps as shown in the figure.

6.1 Histogram intersection

A histogram difference value HD_i (difference between *i*th and (*i+1*)th frame is computed using normalized histogram intersection as follows:

$$HDi = 1 - (1/3\mathrm{n}) \times [\Sigma_{j=1}^{n} min(F_{rj}^{i}, F_{rj}^{i+1}) + \Sigma_{j=1}^{n} min(F_{gj}^{i}, F_{gj}^{i+1}) + \Sigma_{j=1}^{n} min(F_{bj}^{i}, F_{bj}^{i+1})] \quad (6.1)$$

where n is the number of pixels in the frame, and F_{rj}^{i} is the number of *j*th bin of the red plane of the *i*th frame. Similar terms are defined for green and blue planes. This measure ensures that frame which are nearly similar, HD_i turns out to be close to zero, while for dissimilar frame HD_i is closer to one.

Once we have obtained the frame using histogram intersection we can use the local adaptive regression kernel to identify the image in the frame. We apply this method on the whole frame sequence so that we can identify the frame break in video and could match the frame for possible relevance with the query object.

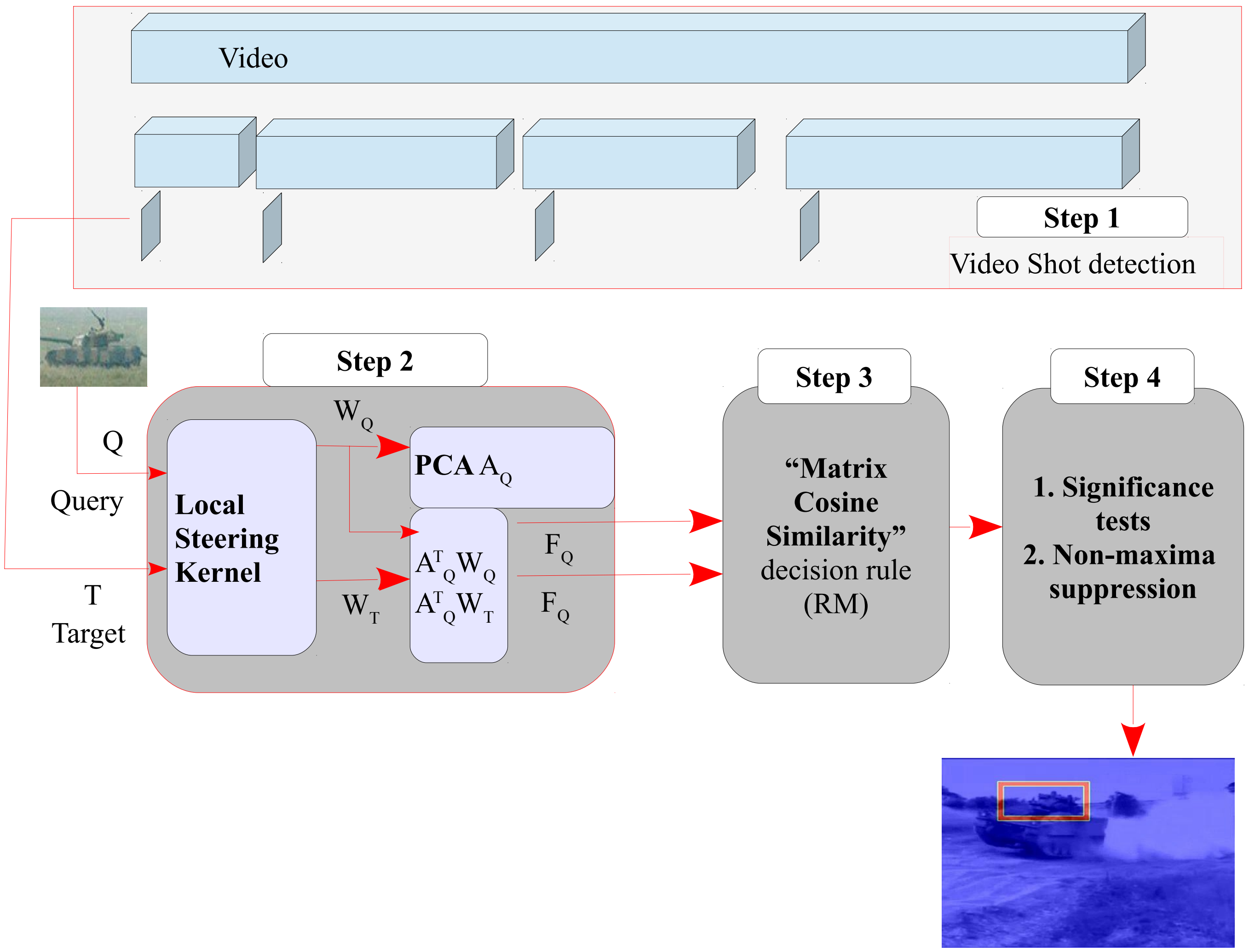


Figure 6.1 System Overview

6.2 Locally Adaptive Kernel Regression

In kernel regression, we made use of spatial differences to weigh the input values. In locally-adaptive kernel regression, we not only make use of spatial differences, but also the difference in data (pixel gradients). In particular, in steering kernel regression, this is done by setting the smoothing matrix H_i to be

$$H_i = hC_i^{1/2} \tag{6.2}$$

where h is a global smoothing parameter and C_i is the covariance matrix at the *i*th pixel, An estimate of this covariance matrix can be obtained using the following formula:

$$C_i = \begin{bmatrix} \Sigma_{x_j \epsilon w_i} f_x(x_j) f_x(x_j) & \Sigma_{x_j \epsilon w_i} f_x(x_j) f_y(x_j) \\ \Sigma_{x_j \epsilon w_i} f_y(x_j) f_x(x_j) & \Sigma_{x_j \epsilon w_i} f_y(x_j) f_y(x_j) \end{bmatrix} \tag{6.3}$$

where f_x and f_y are the derivatives along the x and y directions, and wi is a window surrounding the pixel. If we choose our kernel function to be a Gaussian kernel, then the local steering kernel (LSK) at a pixel x_i will now be given by:

$$K(x_i - x; H_i) = \frac{\sqrt{det(C_i)}}{2\pi h^2} \exp\left(\frac{-(x_i - x)^T C_i (x_i - x)}{2h^2} \right) \tag{6.4}$$

Because the smoothing matrix is now a function of the local pixel data (represented by the covariance matrix), this has the effect of spreading the kernel along local edges. Figure 6.2 shows how a Gaussian kernel adapts to the image data inside the red box.

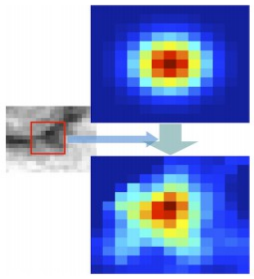


Figure 6.2 Steering Kernel

Object Detection Using Local Steering Kernels Local steering kernels represent the local structures in images, and give us a measure of local pixel similarities. Given a query image Q, target image T, overall threshold τ_0, and a window size P^2, the generic object detection algorithm involves the following:

First, the LSKs for the (grayscale) target and query images must be computed. Let these be denoted by $K_Q(x_i - x; H_i)$ and $K_Q^T(x_i - x; H_i)$, where the subscripts Q and T denote the LSKs for the query and target, respectively, and the superscript j denotes that the kernels were computed for the j_{th} patch in T that is the same size as Q. Now, the LSKs are too dense to use as effective descriptors, and so the next step would be to reduce the dimensionality of these vectors. Before this, however, we first need to normalize our data.

Normalization of $K_Q(x_i - x; H_i)$ and $K_Q^T(x_i - x; H_i)$ is given by the following formulas

$$W_Q(x_i - x) = \frac{K_q(x_i - x; H_i)}{\Sigma_{l=1}^{P^2} K_Q(x_l - x; H_l)} \tag{6.5}$$

$$W_T^j(x_i - x) = \frac{K_T^j(x_i - x; H_i)}{\Sigma_{l=1}^{P^2} K_T^j(x_l - x; H_l)} \tag{6.6}$$

Let W_Q and W_j^T denote the collection of normalized LSKs for all the x_i*'s* in the query, and all the x_i*'s* in the j_{th} patch in the target, respectively. After normalization, Principal Component Analysis (PCA) can then be used to reduce the dimensionality. Applying PCA to W_Q and extracting the top d eigenimages gives us the collection of eigenimages, F_Q, and the projection space A_Q that was used to obtain these eigenimages. We then project W_j^T onto A_Q to obtain F^T, which is a a collection of eigenimages for the target that are in the same space as that of the query.

With these reduced descriptors, we can now compute the similarity between two patches. For this, the Cosine Similarity measure is used. The similarity of the j_{th} patch in the target to the query is given by

$$\rho_j = \langle \frac{F_Q}{\|F_Q\|}, \frac{F_Q}{\|F_Q\|} \rangle \tag{6.7}$$

From this measure, we generate the resemblance map by calculating the resemblance

$$f(\rho_j) = \frac{\rho_j^2}{1 - \rho_j^2} \tag{6.8}$$

for all patches in the target. Finally, significance tests and non-maximum suppression are applied to find the objects. First, the resemblance map is thresholded by the overall threshold τ_0 (ideally set to 0.99), to determine if there are any objects present in the target. If no values are above τ_0, then no objects are present. Then, the resemblance map is thresholded by a second threshold, τ, which is extracted from the PDF of $f(\rho_j)$ and is set so that only 1% of the resemblance values are above it. This gives a 99% confidence level in the produced data. The last step is to apply non-maximum suppression to find the locations of the objects.

We determine the result of the video for abrupt changes in frames by histogram intersection and we see from the figure (3), that it shows the number of the frame at which the change is happening, which we captured and applied to LARK.

We have summarized the above procedure in the form of a pseudo-code which shows the step by step procedure to find out the steering kernel and then detection of the object.

Algorithm 5.1- Pseudo-code for the non-parametric object detection algorithm

```
Q : Query image, T : Target image, τo : Overall threshold, α : Confidence level, P2: Size
of local steering kernel (LSK) window.
Step 1: Histogram intersection
Apply the histogram intersection and find the videoshot in the movie.

Step 1: Feature representation
1) Construct $W_Q$, $W_T$ which are a collection of normalized LSK associated with $Q, T$.

2) Apply PCA to $W_Q$ and obtain projection space $A_Q$ from its top d eigenvectors.

3) Project $W_Q$ and $W_T$ onto $A_Q$ to construct $F_Q$ and $F_T$.

Step 2: Compute Matrix Cosine Similarity
for every target patch $T_i$ , where i ∈ [0, · · · , $M-1$] do
        $\rho_i = \langle \frac{F_Q}{\|F_Q\|_F}, \frac{F_Q}{\|F_Q\|_F} \rangle$ F and compute resemblance map (RM):
    $f(\rho_i) = \frac{\rho_i^2}{1-\rho_i^2}$
.end for
Then, find max $f(\rho_i)$.
Stage3: Significance tests and Non-maxima suppression
```

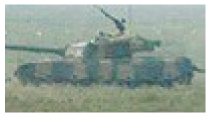
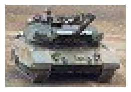
```
1) If max $f(\rho_i) > \tau_o$, go on to the next test. Otherwise, there is no object of interest in $T$.

2) Threshold RM by $\tau$ which is set to achieve 99 % confidence level ($\alpha = 0.99$) from the empirical PDF of $f(\rho_i)$.

3) Apply non-maxima suppression to RM until the local maximum value is below $\tau$.
```
Table 5.1: Pseudo-code for training free videoshot object detection

Chapter 7

Simulation and Results

7.1 Result

We take an image of a 'Tank' as query and try to detect the tank in the video which consist of tanks trials mixed with other images so that we could test our algorithm if it works or not. Figure 7.1 shows the tank query image, and figure 7.2 shows the output of the histogram intersection graphically which shows that at which frame there is an abrupt change in the video. Images of videoshot frames that are arranged in an image matrix form in figure 7.3.

(a) **(b)**

Figure 7.1 Query image

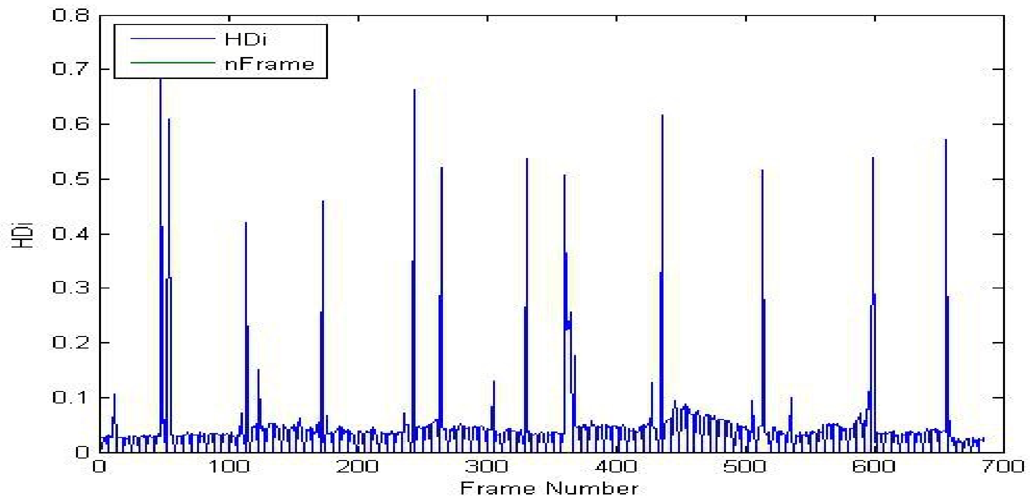


Figure 7.2: Histogram instersection graphical output showing the frame at which abrupt change occurs.

These detected shots are the input to the system for steering kernel matrix calculation. Once these steering kernel matrix is obtained then by going along the algorithm 6.1 we capture the object in these shots and validate our results. Result of above system is again shown in the image matrix form in figure 7.3 which shows the tank detected in each frame.

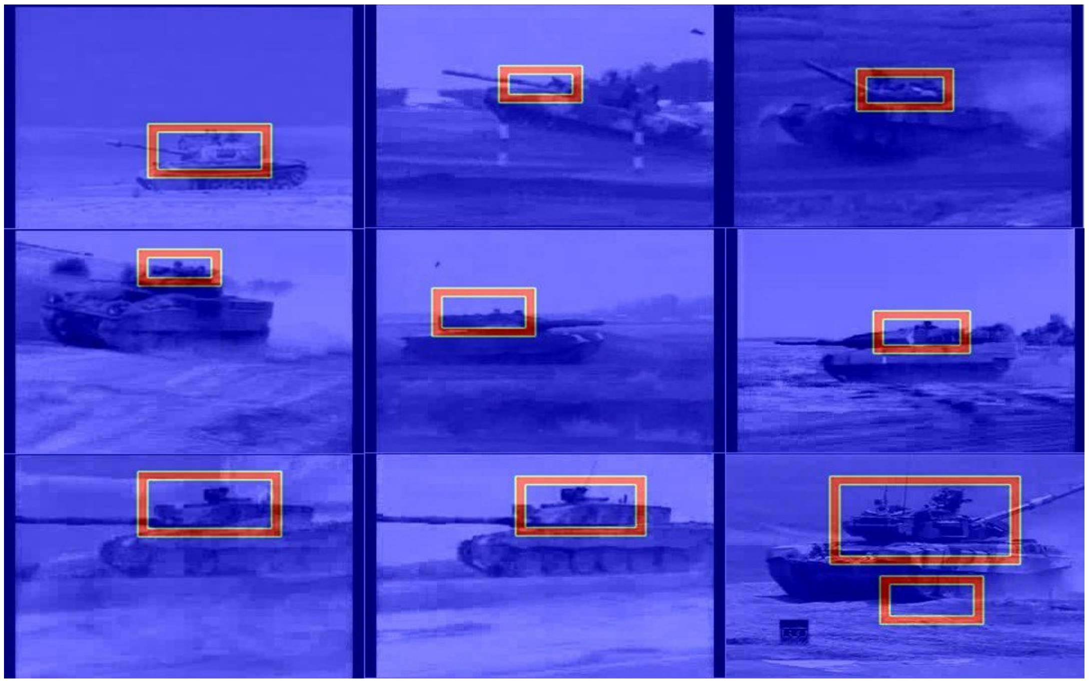


Figure 7.3: Tanks detected in the system in the frames.

7.2 Limitation

Since the above procedure is training free in which we detect the object on the basis of only one query so system has an inherited problem of not being able to detect the same object in case the object is tilted or oriented. For example an image of figure 7.1 is unable to detect the tank as in figure 7.4. The solution for this problem is to use chained query image generation[16] which take all the positive images that query 1 failed to classify in first training set, and use these to generate a second query image. This second query will now have information that contains more details that the first query, since it is averaging over a smaller subset of images. These two queries are then used to perform classification on Training Set.

As we see from the simple example of above procedure in which we use two query images of tanks and for target we take a single image which contains many tanks in different poses. By running the first query in the system we see that the system has detected the tanks in the target image but there are many tanks which are still left undetected. On the second iteration of the system, the system uses the second query image to work on the target image and detects most of the rest of the tanks as we can see in the figure 7.4.

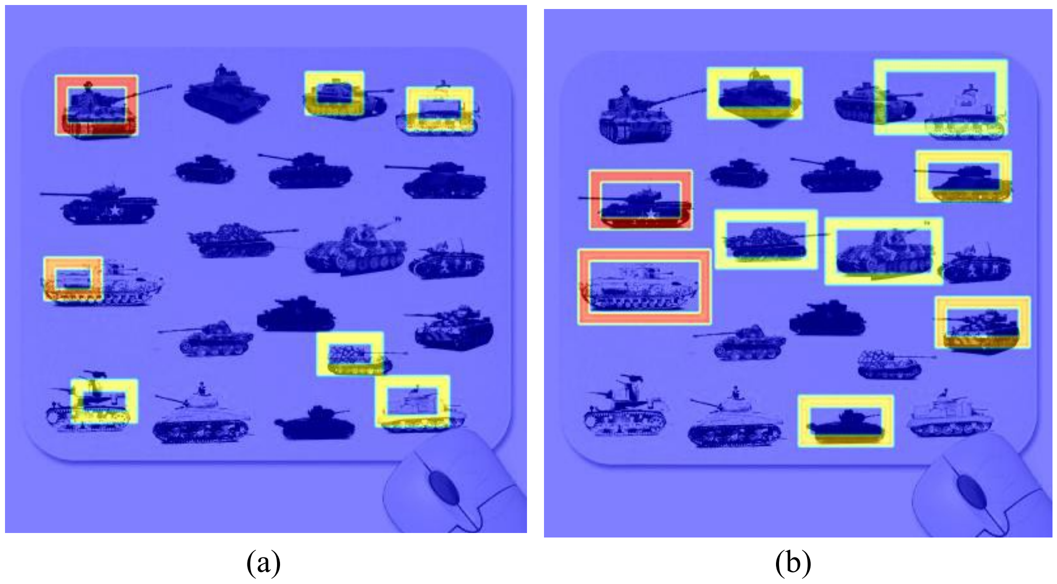


Figure 7.4: Chained query image generation (a) shows the detection using the image of figure 7.1(a) and 7.4(b) shows the detection using the image of figure 7.1(b)

7.3 Conclusions

In this thesis, we studied a non-parametric kernel regression (KR), proposed a novel and powerful training-free non-parametric object detection framework by employing local steering kernels (LSKs) which well capture underlying data structure, and by using the "Matrix Cosine Similarity" (MCS) measure. The proposed method can automatically detect in the target the presence, the number, also the location of objects. We applied the proposed approach for indexing the data from a random video. The experimental results of both simulated and real data showed that the proposed method was effective in indexing the contents of video without any prior knowledge of data to be indexed.

However the success of the indexing is limited to the type of data present in the video itself. An object is successfully indexed when it occupy a small portion the frame, because this helps in identifying the underlying feature of the target object which could be matched against the matrix kernel of query. In case the object occupies the larger portion of the frame then the reliability of the algorithm decreases, but it still tries to find the closest match for the query input.

The proposed method does not require any prior knowledge (learning) about actions being sought; and does not require any segmentation or pre-processing step of the target.

Appendix A

Equivalent Kernels

Study of (1.33) shows that $\mathbf{X}^T\mathbf{KX}$ is an (N + 1) × (N + 1) block matrix, with the following structure:

$$X^T KX = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots \\ a_{21} & a_{22} & a_{23} & \cdots \\ a_{31} & a_{32} & a_{33} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{A.1}$$

where a_{lm} is an $l \times m$ matrix for the 2-D case. The block elements of (A.1) for orders up to N = 2 are as follows:

$$a_{11} = \sum_{i=1}^{P} K_H(x_i - x), \tag{A.2}$$

$$a_{12} = a_{21}^T = \sum_{i=1}^{P} (x_i - x)^T K_H(x_i - x), \tag{A.3}$$

$$a_{22} = \sum_{i=1}^{P} (x_1 - x)(x_1 - x)^T K_H(x_1 - x), \tag{A.4}$$

$$a_{13} = a_{31}^T = \sum_{i=1}^{P} vech^T\left\{(x_1 - x)(x_1 - x)^T\right\} K_H(x_1 - x), \tag{A.5}$$

$$a_{23} = a_{32}^T = \sum_{i=1}^{P} (x_1 - x)\, vech^T\left\{(x_1 - x)(x_1 - x)^T\right\} K_H(x_1 - x), \tag{A.6}$$

$$a_{33} = \sum_{i=1}^{P} (x_1 - x)\, vech\left\{(x_1 - x)(x_1 - x)^T\right\} vech^T\left\{(x_1 - x)(x_1 - x)^T\right\} K_H(x_1 - x), \tag{A.7}$$

With the above shorthand notations, the equivalent kernel functions $W_i(\cdot)$ in (1.35) for up to $N = 2$ are given by

$$W_i(K, H, N=0, x_i - x) = \frac{K_H(x_i - x)}{a_{11}} \qquad \text{(A.8)}$$

$$W_i(K, H, N=0, x_i - x) = \frac{\left\{1 - a_{12} a_{22}^{-1} (x_i - x)\right\} K_H(x_i - x)}{a_{11} - a_{12} a_{22}^{-1} a_{21}} \qquad \text{(A.9)}$$

$$W_i(K, H, N=0, x_i - x) = \frac{\left[1 - A_{12} a_{22}^{-1} (x_i - x) - A_{13} A_{33}^{-1} vech\left\{(x_1 - x)(x_1 - x)^T\right\}\right] K_H(x_i - x)}{a_{11} - a_{12} a_{22}^{-1} a_{21} - A_{13} A_{33}^{-1} a_{31}}$$

with

$$A_{12} = a_{12} - a_{13} a_{33}^{-1} a_{32,} \qquad A_{22} = a_{22} - a_{23} a_{33}^{-1} a_{32,}$$

$$A_{13} = a_{13} - a_{13} a_{22}^{-1} a_{23,} \qquad A_{22} = a_{22} - a_{32} a_{22}^{-1} a_{23,} \qquad \text{(A.10)}$$

Appendix B

Local Gradient Estimation

In this appendix, we formulate the estimation of the direct gradient β_1 of the second order kernel regressor (N = 2). Note that direct gradient estimation is useful not only for the iterative steering kernel regression, but also for many diverse applications such as estimating motion via gradient-based methods (e.g., optical flow) and dominant orientation estimation. In the solution of the optimization of kernel regression (1.32), the second and third element of b give the estimate of local gradient:

$$\nabla \hat{z}(x) = \hat{\boldsymbol{\beta}}_1 = \begin{bmatrix} e_2^T \\ e_3^T \end{bmatrix} \left(\boldsymbol{X}^T \boldsymbol{K} \boldsymbol{X} \right)^{-1} \boldsymbol{X}^T \boldsymbol{K} \boldsymbol{y} \qquad \text{(B.1)}$$

where $\mathbf{e}_2$ and $\mathbf{e}_3$ are column vectors with the second and third elements equal to one, and the rest equal to zero. Using the notations of (A.3)-(A.7) in Appendix A, the local quadratic gradient estimator is expressed as

$$\nabla \hat{z}(x) = \sum_{i=1}^{P} \mathrm{B}^{-1} \left[-B_{21} B_{11}^{-1} + (x_i - x) - B_{23} B_{33}^{-1} vech\left\{ (x_i - x)(x_i - x)^T \right\} \right] K_H(x_i - x)\, y_i, \quad \text{(B.2)}$$

where

$$B_{11} = a_{11} - a_{13} a_{33}^{-1} a_{31}, \qquad B_{21} = a_{21} - a_{22} a_{33}^{-1} a_{31,}$$

$$B_{23} = a_{23} - a_{21} a_{11}^{-1} a_{13}, \qquad B_{33} = a_{33} - a_{31} a_{11}^{-1} a_{13,}$$

$$\mathrm{B} = a_{22} - B_{21} B_{11}^{-1} a_{12} - B_{23} B_{33} - 1\, a_{32.} \qquad \text{(B.3)}$$

References

1. Hideyuki Tamura, Shunji Mori, Takashi Yamawaki, “Textural Features Corresponding to Visual Perception,” IEEE transactions on System Man and Cybernatics, vol. 8, issue 6, pages 460-473.

2. Tushabe, F.; M.H.F. Wilkinson (2008). "Content-based Image Retrieval Using Combined 2D Attribute Pattern Spectra". *Springer Lecture Notes in Computer Science*.

3. Dr Kekre H.B, Mishra D, (2011), Content Based Image Retrieval using Density Distribution and Mean of Binary Patterns of Walsh Transformed Color Images, Vol 3, No 2.

4. Dr Kekre. H.B, Thepade S.D., Maloo A., (2010), CBIR Feature Vector Dimension Reduction with Eigenvectors of Covariance Matrix using Row, Column and Diagonal Mean Sequences, Vol (3)-12, published in International Journal ofComputer Applications (0975 – 8887).

5. Dr Kekre H.B, Mishra D, (2010), Performance Comparison of Four, Eight & Twelve Walsh Transform Sectors Feature.

6. Hae Jong Seo, Peyman Milanfar, "Training-Free, Generic Object Detec-tion Using Locally Adaptive Regression Kernels," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1688-1704, Aug. 2010, doi:10.1109/TPAMI.2009.153.

7. R. S. Jadon, Santanu Chaudhury, K. K. Biswas, “A fuzzy theoretic approach for video segmentation using syntactic features,”Elsevier Pattern Recognition Letters 22 (2001) 1359 -1369.

8. H. Takeda, S. Farsiu, and P. Milanfar, “Kernel Regression for Image Processing and Reconstruction,” IEEE Transactions on Image Processing, vol. 16, no. 2, pp.

349-366, Feb. 2007.

9. “Deblurring using regularized locally-adaptive kernel regression,” IEEE Transactions on Image Processing, vol. 17, pp. 550–563, April 2008.

10. M. P. Wand and M. C. Jones, Kernel Smoothing, ser. Monographs on Statistics and Applied Probability. London; New York: Chapman and Hall, 1995.

11. M. G. Schimek, Smoothing and Regression -Approaches, Computation, and Application-, ser. Wiley Series in Probability and Statistics. New York: Wiley-Interscience, 2000.

12. I. Jolli®e. Principal Component Analysis. Springer-Verlag, New York, 1986.

13. T. Cox and M. Cox. Multidimensional Scaling. Chapman Hall, Boca Raton, 2nd edition, 2001.

14. A. HyvÄarinen. Survey on independent component analysis. Neural Computing Surveys, 2:94-128, 1999.

15. H. Hotelling. Analysis of a complex of statistical variables into components. J. Of Educational Psychology, 24:417-441, 1933.

16. Arthur Louis Alaniz II, Christina Marianne G. Mantaring, “Using Local Steering Kernels to Detect People in Videos”.