NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Thesis Title: Parameter estimation and target classification with a deformable template.

Author: Roger Dufour.

Department: Electrical and Computer Engineering.

Approved for Thesis Requirements of the Doctor of Philosophy Degree:

Thesis Advisor: Prof. Eric Miller	Date
Thesis Committee: Prof. Dana Brooks	Date
Thesis Committee: Prof. David Castañon	Date
Thesis Committee: Dr. Michael McCormack	Date
Department Chair: Prof. Fabrizio Lombardi	Date
Director of the Graduate School: Prof. Yaman Yener	Date

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Thesis Title: Parameter estimation and target classification with a deformable template.

Author: Roger Dufour.

Department: Electrical and Computer Engineering.

Approved for Thesis Requirements of the Doctor of Philosophy Degree:

Thesis Supervisor: Prof. Eric Miller	Date
Thesis Committee: Prof. Dana Brooks	Date
Thesis Committee: Prof. David Castañon	Date
Thesis Committee: Dr. Michael McCormack	Date
Department Chair: Prof. Fabrizio Lombardi	Date
Director of the Graduate School: Prof. Yaman Yener	Date

Copy Deposited in Library:

Reference Librarian	Date

PARAMETER ESTIMATION AND TARGET CLASSIFICATION WITH A DEFORMABLE TEMPLATE

A Thesis Presented

by

Roger Dufour

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the field of

Electrical Engineering

Northeastern University

Boston, Massachusetts

May 2003

Preface

Two common image processing problems are determining the location of an object using a template when the size and rotation of the true target are unknowns and classifying an object into one of a library of objects again using a template-based matching technique. When employing a maximum likelihood approach to these problems, complications occur due to local maxima on the likelihood surface. In this thesis, we have demonstrated a technique for object localization that employs a library of templates ranging from a smooth approximation template to the exact template with varying degrees of detail. Successively estimating the geometric parameters (i.e. size and rotation) using these templates achieves the accuracy of the exact template while remaining within a well-behaved "bowl" in the search space which allows standard maximization techniques to be used. Further, this technique is extensible to solve the classification problem using a multiple template library. We introduce a steering parameter that at every scale, allows us to compute a template as a linear combination of templates in the library. The algorithm begins the template matching using a smooth blob which is the smooth approximation common to all templates in the library. As the location and geometric parameter estimates are improved and detail is added, the smooth template is "steered" towards the most likely template in the library and thus classification is achieved. In this thesis, we have developed these algorithms and demonstrated their performance against both simulated data and real data.

Acknowledgements

Perhaps the greatest pleasure of finishing any work is thanking those who aided in its completion. Indeed, without a great many people this thesis could never have been finished. Foremost, I must thank Professor Eric Miller without whose help and advice this thesis would not be possible. His knowledge and guidance were endless. I enjoyed working with Eric throughtout both my masters and doctoral research. I must also thank my committee members, Professors Dana Brooks and David Castañon and Dr. Michael McCormack. They made many helpful suggestions in the final writing of this thesis and it is a better work because of them. I would like to thank my family, friends and loved ones who have always supported me throughout this process, especially my mother and father.

Contents

List of Figures

Chapter 1

Introduction

Two common image processing problems are determining the location of an object using a template when the size and rotation of the true object are unknowns [1, 5, 25] and classifying an object into one of a library using a template-based matching technique [9, 28, 37]. It is desirable that an algorithm for finding the solution to either problem should be robust to noise, accurate across a wide range of object configurations, and computationally efficient. Because of the added computational complexity involved in the target classification problem, it is often treated separately and employs techniques which treat the size, rotation and location as nuisance parameters [9, 28, 37]. In this thesis, we present a unified approach to these problems and show that the target class parameter can be treated equivalently with the geometric parameters and that determination of all the parameters can improve performance for both localization and classification.

For the object location problem with known geometric parameters (i.e. size and rotation) and additive Gaussian noise, the classic solution is a whitening filter followed by a matched filter. This solution produces the maximum output SNR, but the resulting signal often has a broad peak reducing confidence in the location accuracy. Further, this filter is highly energy dependent and often will produce a significant response to areas of high amplitude clutter. Other estimators have been proposed, such as the phase only matched filter (POMF) and the symmetric phase only matched filter (SPOMF) which give better location discrimination than the standard matched filter [25]. However these filters do not use additional information regarding the background noise which may be available or estimable. Inclusion of these statistics can overcome problems associated with clutter or colored noise and thus provide a more accurate location estimate. Alternatively, one may formulate the localization problem in the framework of image reconstruction, where the image to be recovered is a delta function at the location of the object and the blurring function is the template [1]. This approach makes available image reconstruction methods such as the linear least squares estimator (LLSE) and maximum likelihood estimators which can incorporate background statistics and provide a solution which is optimal with respect to maximizing the posterior probability density function of location conditioned on the data.

Since they rely upon an accurate template, the localization methods mentioned above are not sufficient when geometric parameters such as size and rotation are not known. With unknown geometric parameters, two approaches are possible. The first is to treat the geometric parameters as nuisance parameters and design a filter which is invariant to these, and the second is to simultaneously estimate the geometric parameters and the location parameters. With the first approach, accurate solutions are possible with computationally efficient estimators since we are generally only processing the data once. However, the use of invariance means that significant information related to the geometric parameters is being discarded. Further, in such applications as tracking, it is often desirable to have the geometric information to aid in estimating the object's motion. With the second approach, simultaneous estimation, the geometric parameters are also provided usually at the expense of computational efficiency and a more complex algorithm. One example of this approach is the use of the Fourier-Mellin (FM) transform. The FM transforms the scale and rotational parameters to translational parameters, which can be estimated using one of the above mentioned methods [5]. The transform is invariant to the location parameters. Thus the geometric parameters are first estimated in the FM domain, then a matched filter, with the proper parameters, is used for the location estimate.

As an alternative approach, one could use maximum likelihood estimation of the scale, rotation and location parameters directly. This would be highly accurate and would have the advantage of being optimal with respect to maximizing the posterior probability. To perform this estimation, typically, one would hope to find a closed form solution for the maximum of the likelihood surface. In general this is not possible so one could, alternatively, use some optimization routine to find this maximum. However, with the problem at hand, this approach is not feasible since the likelihood surface (as we will demonstrate) has numerous local maxima and areas of zero gradient and is thus not amenable to a hill climbing algorithm which would likely become trapped at a local maxima. A solution of last resort could of course be achieved by calculating the likelihood value for a densely sampled grid of parameters. However, this exhaustive search is generally too costly computationally. In this thesis, we will present a method for maximizing this surface by operating upon better behaved approximations to the surface computed using approximations to the template.

In the first part of this thesis, we demonstrate the method for searching this surface using a progression of templates. The early templates are smooth monomodal approximations of the exact template. This allows us to search a well-behaved approximation of the true likelihood surface. Here we can use a standard optimization routine such as the Newton algorithm to find the best fit solution. Using this estimate as the starting point of the next estimation, we add detail to our template and search again. As we add detail, the surface becomes more ill-behaved. The local maxima observed on the true likelihood surface begin to appear, but the previous estimates have placed us within a better behaved "basin of attraction" of the global maximum. By adding detail slowly, our successive estimates of the maximum will hopefully remain with this basin, until adding detail to the approximate template produces the full-detail template and the estimation settles at the maximum likelihood value of the parameters.

To generate the templates for the search, we first must recognize the desirable qualities for the template approximation routine: the early templates must be smooth and monomodal, the progression must be continuous so steps can be arbitrarily small, and all approximate templates must be continuous. We found we could achieve all these conditions by using a diffusion-like equation which provides all the intermediate templates between the smooth approximate template and the full-detail template. The diffusion equation also allows fast Fourier based computations of the templates. Coupled with the Fourier based image reconstruction method [1], this leads to a Fourier algorithm which is not computationally burdensome.

Akin to the parameter estimation problem is the target classification problem. With target classification, the objective is to make a determination, based upon the data, as to which target is represented. The maximum likelihood solution to this problem in the presence of Gaussian noise would be achieved by comparing the data to the ideal target templates matched in size and rotation, and selecting the one with the least squared difference. However, parameter estimation would necessarily complicate this setup since the true parameters would likely not be available. Alternative solutions have been developed to overcome this difficulty. Many methods of target classification seek to develop features which are invariant to size and rotation. Classification is then performed in the feature space. However, by casting these parameters as nuisance parameters, information is necessarily destroyed in developing the features.

In the second part of this thesis, we extend the template matching method to the problem of target classification. To overcome the local minima problems associated with the geometric parameters we introduced a smoothing operation on the template. With properly normalized templates, this smoothing operation will result in all template estimation in the library beginning with an identical Gaussian blob. Now, as we add detail, we must make a decision as to which template in the library we are evolving towards. We could, at some point of smoothness, choose the template which achieves the highest likelihood score. This would be a hard decision and would necessitate discarding the other templates as we pursued the most likely choice. Alternatively, we could make a soft decision and assign more weight to the likely template while still carrying forward the other, less likely, candidates which may in fact have the correct template among them. The method for treating the several templates is to compute a meta-template which is a linear combination of the template in the library, weighted by the posterior probability of that template given the data and the current estimate of the geometric parameters. The vector of weights allows the meta-template to be steered through a continuum of templates defined by the template library. As detail is added, we can steer this template towards the most likely template in the library, but we delay making any hard decision until the geometric parameter estimates become more accurate. This is the method which we use for classification. We have introduced a steering vector which selects from a continuum of templates. Now as we estimate our geometric parameters, we also resolve the target classification. As detail is added to the templates, the steering vector will move towards the template which best approximates the data.

The remainder of the introduction is organized into three sections. In the first we will present the problem formulation with respect to the data which we have available, the templates, and the parameters which we will estimate. We will also show the likelihood equations and an example of the likelihood surface which was discussed above and which motivates the work of the thesis. The second section will give a brief overview of the contributions of this thesis. The third section will outline the organization of the remainder of the thesis.

1.1 The Problem Formulation

The typical imaging system which we are examining in this thesis will produce a pixilated image of the object of concern against a background with additive noise. Throughout this work, we will typically use a continuous formulation of the images, reducing to the sampled, or pixilated, images only for implementation purposes. We should first describe the data as

$$g(\mathbf{r}) = f_{E,k^0}(\mathbf{r} - \bar{\mathbf{r}}^0; \boldsymbol{\theta}^0) + n(\mathbf{r}), \tag{1.1}$$

where $\mathbf{r}$ is the two-dimensional vector of coordinates in the image space, $g(\mathbf{r})$ is the data, $f_{E,k^0}(\mathbf{r} - \bar{\mathbf{r}}^0; \boldsymbol{\theta}^0)$ is the parameterized template and $n(\mathbf{r})$ is additive noise.

The template in this equation is subscripted by E to express that it is the true or *exact* template as opposed to the approximation templates which will be used later and by k^0 which is the index into the template library. The parameters of this template describe where it is located in the image, $\bar{\mathbf{r}}^0$, and the set of geometric parameters $\boldsymbol{\theta}^0$. The geometric parameters $\boldsymbol{\theta}$ are the size and rotation respectively as

$$\boldsymbol{\theta}^T = [s \ \phi]. \tag{1.2}$$

The template can be computed as a rotated and resized version of the standard template which has only a spatial argument, $f_{E,k}(\mathbf{r})$, given as

$$f_{E,k}(\mathbf{r}; \boldsymbol{\theta}) = f_{E,k}\left(\frac{1}{s}\mathbf{M}(\phi)\mathbf{r}\right), \text{ where } \mathbf{M}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}. \tag{1.3}$$

Throughout this thesis, we will denote the true parameter values with a superscripted zero as was done above. We will denote estimates of the parameters using the hat symbol. Thus, $\hat{\boldsymbol{\theta}}$ would be the estimate for $\boldsymbol{\theta}$, and would hopefully be close to the true value $\boldsymbol{\theta}^0$.

1.1.1 Likelihood Equations

As discussed above, we will use the likelihood setup of the problem to define our cost function. The maximum likelihood equation solution is predicated on maximizing the posterior probability across all possible parameters. Therefore, the likelihood cost function is the posterior probability with the parameters as the independent variables as

$$L(\bar{\mathbf{r}}, \boldsymbol{\theta}, k | g(\mathbf{r})) = p(g(\mathbf{r}) | \bar{\mathbf{r}}, \boldsymbol{\theta}, k). \tag{1.4}$$

In the case of white Gaussian noise, the likelihood function is [11]

$$L(\bar{\mathbf{r}}, \boldsymbol{\theta}, k | g(\mathbf{r})) = C \exp\left(-\frac{1}{2\sigma_n^2} ||g(\mathbf{r}) - f_{E,k}(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta})||^2\right), \tag{1.5}$$

where the norm function $|| \cdot ||$ is the standard 2-norm as

$$||f(\mathbf{r})|| = \left(\int f^2(\mathbf{r})\, d\mathbf{r}\right)^{\frac{1}{2}}. \tag{1.6}$$

It is common, and far simpler in the Gaussian case, to work with the logarithm of the likelihood. Since this is a monotonic transformation is does not change the location of the maximum and it eliminates the exponential. The log-likelihood expression is

$$l(\bar{\mathbf{r}}, \boldsymbol{\theta}, k | g(\mathbf{r})) = \log C - \frac{1}{2\sigma_n^2} ||g(\mathbf{r}) - f_{E,k}(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta})||^2. \tag{1.7}$$

The maximum of this equation is not dependent upon the first term which is a constant. The maximum is solely dependent upon the norm expression and we can define a simpler cost function if we focus on this term alone. Therefore, we can maximize the likelihood by minimizing a cost function defined as

$$J(\bar{\mathbf{r}}, \boldsymbol{\theta}, k | g(\mathbf{r})) \triangleq \frac{1}{2\sigma_n^2} ||g(\mathbf{r}) - f_{E,k}(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta})||^2. \tag{1.8}$$

Thus, the solution to the full problem is the minimization of (1.8) across all possible locations, sizes, rotations, and targets in the library. The expression for our estimates of the geometric parameters, the location and the template (where, e.g. $\hat{\boldsymbol{\theta}}$ is the estimate of the parameter vector $\boldsymbol{\theta}$) is

$$\{\hat{\boldsymbol{\theta}}, \hat{\mathbf{r}}, \hat{k}\} = \arg \min_{\{\boldsymbol{\theta}, \bar{\mathbf{r}}, k\}} J(\bar{\mathbf{r}}, \boldsymbol{\theta}, k | g(\mathbf{r})). \tag{1.9}$$

For the parameter estimation work presented in the first part of this thesis, we assume a single template library. Since k has only a single possible value, we will simplify the cost function in (1.9) to be

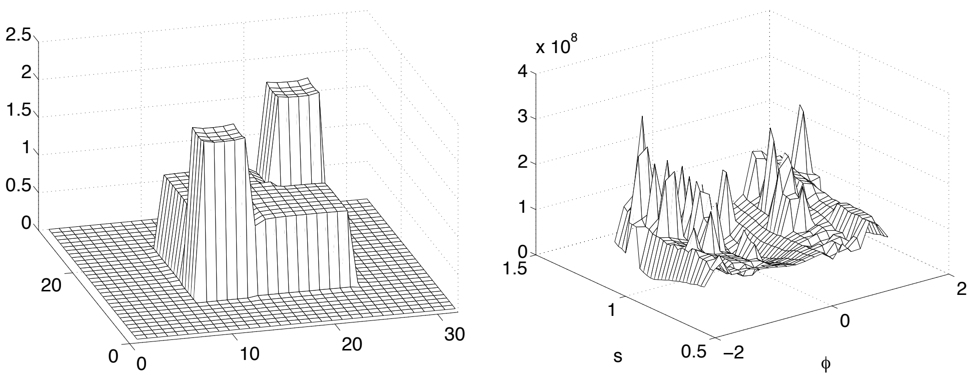
$$\{\hat{\boldsymbol{\theta}}, \hat{\mathbf{r}}\} = \arg \max_{\{\boldsymbol{\theta}, \bar{\mathbf{r}}\}} J(\bar{\mathbf{r}}, \boldsymbol{\theta}; g(\mathbf{r})). \tag{1.10}$$

This expression assumes a proper value of k is used and it is therefore dropped from the expression. In the following subsection we will use the single template expression for examining the difficulties associated with minimizing (1.9).

1.1.2 Complications of the Likelihood Solution

If it were possible to find a closed form solution for the minimum in (1.10), then the problem would be trivial for both estimation and classification since we could compute the estimate and cost value for each template in the library and select the lowest cost solution. Alternatively, barring computing the minimum directly, we could minimize on a cost surface generated with each template and again select the lowest cost solution. However, as discussed earlier the surface to be minimized is not amenable to a gradient descent minimization due to numerous local minima and other areas of zero gradient.

An example of the cost surface generated for estimating the size and rotation of a blocky target in noise is shown in Figure 1.1. The template shown in (a) is embedded in noise with 20 dB SNR, a size of 1.0, and a rotation of 0.0. In (b) the cost J is



(a) (b)

Figure 1.1: The two-peak block template and the cost surface with respect to size and rotation.

computed on a grid in $s - \phi$ space. The global minimum is correctly found to be at a value of $s = 1.0$ and $\phi = 0.0$. However, minimization would be complicated by the local minima and the general spiky nature of the surface. These difficulties motivate the search for an alternative minimization method which will be examined in more depth in Chapter 3.

1.2 Summary of Contributions

In this thesis, we explored the problems of geometric parameter estimation and target classification. We composed these two problems as template matching problems and developed a common method for their solution. We showed that the problems can be placed in a consistent setting.

In the first part of this thesis we examined the problem of target parameter estimation. We created an algorithm for solving for the geometric parameters of size and rotation in addition to the location of the target. The algorithm presented is an adaptation of template matching which incorporates template smoothing to overcome the ill-behavior of the likelihood surface. In this thesis, we developed the specifics of the algorithm: the template progression, the parameter optimization, and the scheduling of the smoothed templates. We demonstrated this algorithm versus real and synthetic data and compared the performance to the Fourier Mellin Matched Filter performance.

In the second part of this thesis, we extended the parameter estimation algorithm to target classification. This was done by showing that the degenerate template for any target can be identical. We then developed a continuous template model which can be "steered" to any of the canonical templates in the library or any combination of them. Then the classification algorithm is developed by creating a steerable template, a template progression for the steerable template, and an optimization step for the additional parameters. The parameters of the steerable template can live either on a simplex, or on a Stiefel manifold. We showed the advantages of both and developed optimization algorithms for both surfaces. Further, we developed a theory which relates the final estimated values of the steerable template to the posterior probability mass function of the templates in the template library. Lastly, we added additional cost penalties to the surface to impose better solutions.

1.3 Organization of the Thesis

In Chapter 2 of this thesis we examine some background research and related methods for performing template matching, target parameter estimation and target classification. We will examine more extensively the work of Abu-Naser et. al. which we use to decouple the estimate of location from the estimate of the geometric parameters. In Chapter 3 we examine our method for performing the parameter estimation using a template smoothing algorithm. We will examine the performance of this algorithm versus real and synthetic data in Chapter 4. We continue the work on the template smoothing algorithm by extending it to the problem of target classification in Chapter 5. In Chapter 6 we will examine the performance of a series of algorithms which were developed. Lastly, Chapter 7 discusses some conclusions and expected future work.

Chapter 2

Background

The work of the thesis is divided into two fields, that of detection of a template in an image and that of classification. The review of methods follows this division. In the first part we discuss template matching methods of two types, those methods which rely upon known geometric parameters and those methods which estimate the geometric parameters. The second part discusses classification methods which all work with unknown geometric parameters.

2.1 Template Matching Techniques

Template matching techniques at their most basic level, search an image for the best fit of the template [1, 2, 5, 17, 34]. The fit can be scored by correlation methods or a two-norm of the difference between the data and the estimated template, or some other cost function. The methods proposed seek fast solutions, which are robust to noise and clutter and produce accurate localizations of the target. In the presence of known geometric parameters, we will see that matched filters and variations thereof are the preferred methods, but that these techniques are not sufficient when these parameters are unknown.

2.1.1 Known Geometric Parameters

In the presence of known geometric parameters, i.e. size and rotation, the solution can be regarded as the best correlation between the template and the data [1,12,16,17, 21,22]. One could compute a correlation for many points in space and select the point of highest response. Indeed, it can be seen that the matched filter is this solution and, under white Gaussian noise conditions, this produces the highest output SNR between the signal and the background. However, the matched filter is generally a low-pass process and so produces a broad peak which is not suited to precise target localization. Other solutions which produce higher resolution in the localizations have been proposed, such as the phase only matched filter or filters constructed using image reconstruction techniques. These filters often produce sharper location estimates or are more robust to noise.

Matched Filter Techniques

The most common and obvious technique to perform template matching, at least with known geometric parameters, is to use a matched filter [17]. Letting $s = 1$ and $\phi = 0$, the matched filter is then composed simply as the reversed template with a whitening filter, giving the transfer function [17]

$$\tilde{h}(\mathbf{k}) = \frac{\tilde{f}_0^*(\mathbf{k})}{S_n(\mathbf{k})}. \tag{2.1}$$

The location is then selected as the maximum output of this filter

$$\hat{\mathbf{r}} = \arg\max_{\mathbf{r}} h(\mathbf{r}) * g(\mathbf{r}) = \arg\max_{\mathbf{r}} y(\mathbf{r}). \tag{2.2}$$

which yields the largest SNR achievable with a linear filter for the output signal [17]. However, since the response of this filter is driven primarily by the local energy of the image, it is difficult to differentiate the object from clutter. Further, the structure of $y(\mathbf{r})$ in (2.2) is usually such that it has a broad maximum which is difficult to accurately locate in the presence of noise [5]. For these reasons, it is usually preferable to employ a more sophisticated filter.

Phase Only Matched Filtering

The advantages of phase only matched filters (POMF) over conventional matched filters is the sharpness of the resultant peak and thus the fidelity of the position estimation, and the fact that it is not as significantly effected by the energy of the data [5,16,21]. The transfer function of the phase only matched filter is equal to the phase of the template as [5]

$$\tilde{h}(\mathbf{k}) = \exp[-j\psi_f(\mathbf{k})], \tag{2.3}$$

where $\psi_f(\mathbf{k})$ is the phase of the template spectrum. The application of this filter produces much sharper location estimates than does the matched filter. Also since it is not driven by the local energy, it offers much better discrimination with respect to clutter than does the MF [5].

Further enhancement of the POMF can be achieved by extracting the phase of both the data and the template and using a nonlinear filter to provide the correlation [12]. Also, with real data additional SNR can be achieved with the symmetric phase only matched filter, which exploits the symmetry of the Fourier spectrum [5].

Impulse Reconstruction Techniques

Another technique which produces better localization is the impulse reconstruction technique [1,6,7,18,26]. This technique treats the image data as an impulse convolved with the target with additive clutter and noise. The advantage of this technique is that it applies the vast field of image reconstruction techniques to the problem of target localization.

Following Abu-Naser et. al. [1] the problem of template matching is formulated in an image reconstruction framework, taking a delta function at the template location as the object to be reconstructed and the target template as the blurring kernel, which leads to the convolution equation,

$$g(\mathbf{r}) = f(\mathbf{r}; \boldsymbol{\theta}^0) * \delta(\mathbf{r} - \bar{\mathbf{r}}^0) + n(\mathbf{r}). \tag{2.4}$$

where the $*$ represents two dimensional convolution. The location can now be found by reconstructing the delta function, or inverting the convolution. One possible approach to this could be an inverse filter. However, the convolution tends to be an ill-conditioned matrix, therefore inversion often leads to the image being overwhelmed with high-frequency artifacts. Some type of regularization would be necessary.

In a Gaussian environment, the optimal linear filter is the linear least squares estimate (LLSE) of the image given as [1]

$$\hat{\delta}(\mathbf{r}) = \mathcal{F}^{-1}\left\{ \frac{\tilde{f}_0^*(\mathbf{k}; \boldsymbol{\theta})}{|\tilde{f}_0(\mathbf{k}; \boldsymbol{\theta})|^2 + \left(\frac{\sigma_n}{\sigma_\delta}\right)^2} \tilde{g}(\mathbf{k}) \right\}, \tag{2.5}$$

where σ_n^2 and σ_δ^2 are the noise and signal powers respectively. We can now choose a location estimate by selecting the point of maximum response as our position estimate, as

$$\hat{\mathbf{r}}(\boldsymbol{\theta}) = \arg\max_{\mathbf{r}} \hat{\delta}(\mathbf{r}). \tag{2.6}$$

Abu-Naser et. al. improve this estimation technique further by embedding the estimation in an Expectation Maximization algorithm which allows for estimation of the noise and clutter statistics along with the position estimate [1].

2.1.2 Unknown Geometric Parameters

The preceding techniques are not sufficient when the geometric parameters are unknown. In all cases, in order to perform the location estimation an accurate template is needed. When we lack the geometric information about the template, it becomes necessary to either simultaneously estimate these parameters, or construct techniques which are invariant to them [1, 2, 5, 23].

Fourier-Mellin Matched Filtering

The Fourier Mellin Transform is a transformation of an image such that scale and rotation of the original are mapped into translations of the transform [5, 32]. To compute the FM transform of an image, we first take a Fourier transform, and then remap this Fourier transform using the log-polar coordinates in the frequency domain [5].

Consider an image $g(r_x, r_y)$ which is a resized rotated and translated version of $f(r_x, r_y)$ as [5]

$$g(\mathbf{r}) = f(\frac{1}{s}\mathbf{M}(\phi)\mathbf{r} + \bar{\mathbf{r}}) \tag{2.7}$$

or

$$g(r_x, r_y) = f\left(\frac{1}{s}(r_x \cos\phi + r_y \sin\phi) + \mathbf{r}_x, \frac{1}{s}(-r_x \sin\phi + r_y \cos\phi) + \mathbf{r}_y)\right) \tag{2.8}$$

Then the Fourier transform of this becomes

$$\tilde{g}(k_x, k_y) = e^{-j\psi_f(k_x,k_y)} e^{-j(r_x k_x + r_y k_y)} s^2 \left|\tilde{f}\left(s(k_x \cos\phi + k_y \sin\phi), s(-k_x \sin\phi + k_y \cos\phi)\right)\right| \tag{2.9}$$

where $\psi_f(k_x, k_y)$ is the spectral phase of f. The phase depends upon the translation, rotation and scaling, but we can see that the magnitude is translation invariant as [5]

$$|\tilde{g}(k_x, k_y)| = s^2 \left|\tilde{f}\left(s(k_x \cos\phi + k_y \sin\phi), s(-k_x \sin\phi + k_y \cos\phi)\right)\right|. \tag{2.10}$$

If we now convert to polar coordinates, we can decouple the scale and rotation parameters. Let us define the polar representations of $|\tilde{f}|$ and $|\tilde{g}|$ as [5]

$$\tilde{f}_p(\alpha,\rho) = |\tilde{f}(\rho\cos\alpha, \rho\sin\alpha)| \qquad \tilde{g}_p(\alpha,\rho) = |\tilde{g}(\rho\cos\alpha, \rho\sin\alpha)|. \tag{2.11}$$

This leads to the relation that [5]

$$\tilde{g}_p(\alpha,\rho) = s^2 \tilde{f}_p(\alpha-\phi, \rho s). \tag{2.12}$$

If we remap the scale axis using $\lambda = \log\rho$, and define this as $\tilde{g}_{pl}(\alpha,\lambda) = \tilde{g}_p(\alpha,\rho)$, then we have [5]

$$\tilde{g}_{pl}(\alpha,\lambda) = s^2 \tilde{f}_{pl}(\alpha-\phi, \lambda - \log s). \tag{2.13}$$

Rotation and scaling are now translations. We can now estimate rotation and scaling using a matched filter or POMF in the Fourier-Mellin domain. Chen et. al. [5] found good results using this technique. One of the major drawbacks of this technique however, is that it is sensitive to clutter and to template mismatch.

Geometric Parameters as a Lie Group

An interesting approach to parameter estimation problem is to treat the parameter vectors as elements of a Lie group [15, 33]. It can be shown that the translational and rotational parameters of a template can be treated together, along with a group operation to form the special orthogonal group $SO(n)$, where n is the dimension of the space being operated in. For our problem this would be $SO(2)$. The elements of the group are defined as [33]

$$p = \begin{bmatrix} \mathbf{M}(\phi) & \mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.14}$$

where $\mathbf{M}(\phi)$ is the Given's rotation matrix and $\mathbf{v}$ is an offset vector indicating position. Then we can see that the group operation is matrix multiplication as [33]

$$p_1 p_2 = \begin{bmatrix} \mathbf{M}(\phi_1) & \mathbf{v}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{M}(\phi_2) & \mathbf{v}_2 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M}(\phi_1 + \phi 2) & \mathbf{M}(\phi_1)\mathbf{v}_2 + \mathbf{v}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{2.15}$$

which is an element of the group.

The parameter estimation problem is now a problem of minimizing the cost function on the manifold which represents the Lie group. This is still not an easy problem and retains all the difficulties of local minima and local flatness.

Miller et. al. [27] apply this setup to target tracking by using a jump diffusion method to minimize the surface. By using tracking information, they are able to impose a prior distribution on the surface for the likely minimum. They then estimate a posterior distribution on the surface by stochastically generating samples, computing the costs and updating the model.

2.2 Classification Techniques

Closely related to the template matching techniques are certain target classification techniques. Here we have an image which contains one of several templates in a possible library of templates at an unknown translation, rotation and scaling. The optimal solution from a maximum likelihood approach would be to test all templates in the library at all translation, rotations and scalings. This could possibly be achieved using the previous techniques mentioned for each template, but the cost becomes prohibitive as the size of the library increases.

Affine Transform Invariance

An alternative approach is through the use of an affine invariant transform. Here, a metric is computed from the image which is invariant to translation, rotation and scaling of the object. Invariant techniques can be divided into two categories: local invariants and global invariants. The local invariant functions compute a metric using local features of the object, such as bright spots, lines, etc [9,28,37]. This set of metrics is then compared to a set of metrics for each target and the most appropriate target is selected. This method is particularly useful in the presence of clutter or occlusion of the target. However, this method reduces the information provided by the data significantly.

The global invariant approach attempts to compute metrics from the entire image [3, 35]. These techniques include using moments, or operating in some transform space. The advantage of these techniques is that more information is retained from the data, however these techniques are often not robust to clutter or occlusion. Ben-Arie et. al. [3, 36] present a technique for operating in the Fourier transform domain, where the spectral signature of the template becomes invariant to translation, slant and tilt (for 3D targets). They can then compare this to a library of signatures for each target at several rotations (since the transform is not rotation invariant).

2.3 Optimization Methods

Much of the work of this thesis will focus upon optimization of a cost function. In this section we will briefly touch upon some optimization techniques which could be used for this problem. We will discuss in slightly more depth the Newton algorithm since it will be an essential part of the work in the next chapter.

There are numerous methods for optimizing a function (either maximization or minimization). These can be divided into deterministic techniques which usually rely upon some type of ordered descent search or stochastic techniques which can search the surface in a more haphazard way but have the advantage that they can often escape from local minima which would trap a deterministic minimization.

Common methods in the deterministic descent type algorithms are a gradient descent which uses the first derivative, a Newton method which relies on the first two derivatives, a conjugate direction method, or a combination of these such as the Conjugate Gradient algorithm [8,13,14,31]. On the stochastic search side, the most common method is the simulated annealing method [19].

2.3.1 Descent Methods

Perhaps the simplest of the descent methods would be the conjugate direction method [13]. The conjugate direction method performs line searches along one variable of the function until a minimum is found, it then switches to another variable. This method has the advantage that it is simple, and will converge quickly for well-behaved surfaces when the starting point is near the minimum. However, it will often take a large number of iterations if one of these conditions is not met. Further, the method does not use any information regarding the local gradient or curvature of the cost surface.

The gradient descent method [13] is an alternative approach to the conjugate direction. Here, the search direction is chosen to be in the direction of greatest descent with the assumption that this is likely to be the direction in which the minimum lies. This method performs line searches in the direction of the gradient, then updates the gradient and searches again. No attempt is made to account for the previous path the search has taken, unlike the conjugate direction which is solely based upon this.

A combination of these techniques is the conjugate gradient method [8,13,14]. In the conjugate gradient method, the new search direction is calculated as a combination of the conjugate direction and the present gradient. This has the advantage of incorporating local gradient information, but also preserving a memory of the path taken. By including the conjugacy condition, the search path will tend towards being at a large angle from the previous search direction. This can greatly improve the speed of searches, and the CG algorithm usually converges faster than either of the previous techniques. Further discussion will be given to the CG algorithm shortly.

The Newton Algorithm

The minimization of the geometric parameters in the next section will be done using a Newton algorithm. The Newton algorithm uses the gradient and the Hessian of the function it is minimizing, or the matrices of the first two derivatives. Because the Newton algorithm uses the second derivative information, it can converge much faster than a gradient descent algorithm. Indeed, in the case of a quadratic surface, the Newton algorithm will reach the exact minimum in one step [13].

The Newton algorithm is based on the assumption that the surface can be locally approximated by a quadratic, or a truncation of the Taylor series at the second order term [13]. That is, if we wished to minimize a function $f(\mathbf{x})$ for $\mathbf{x}$, the we should first write the truncated Taylor series as [13]

$$f(\mathbf{x}_k + \boldsymbol{\delta}) \approx q_k(\boldsymbol{\delta}) = f(\mathbf{x}_k) + g^T(\mathbf{x}_k)\boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T G(\mathbf{x}_k)\boldsymbol{\delta}, \tag{2.16}$$

where

$$g(\mathbf{x}_k) = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k} \qquad G(\mathbf{x}_k) = \nabla^2 f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_k}. \tag{2.17}$$

Now in the Newton method, the iterate $\mathbf{x}_{k+1}$ is taken to be

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k, \tag{2.18}$$

where $\boldsymbol{\delta}_k$ is the minimizer of $q_k(\boldsymbol{\delta})$, which is arrived at by

$$G(\mathbf{x}_k)\boldsymbol{\delta}_k = -g(\mathbf{x}_k). \tag{2.19}$$

Thus the Newton iteration step can be performed by: (a) calculating $g(\mathbf{x}_k)$ and $G(\mathbf{x}_k)$, (b) solving for $\boldsymbol{\delta}_k$ using the equation (2.19), and (c) calculating the update of the iterate $\mathbf{x}_k$ using (2.18). It is now evident why the Newton algorithm converges in a single step for a quadratic surface; the approximation $q_k(\mathbf{x})$ is exact and thus the

Newton step minimization is the global minimum of $f(\mathbf{x})$.

Numerous difficulties exist with the Newton method and many algorithms have been implemented to overcome these. Some of the improvements are in refining the search direction or the length of the step taken to ensure convergence or a decrease in the cost function [13]. Other algorithms have examined reducing the Newton algorithms reliance upon the second derivatives which may not always be available. Algorithms such as the Quasi-Newton method and the Newton-Raphson method are often employed instead [8, 13, 14].

2.4 Optimization on a Stiefel Manifold

In the classification work presented in Chapter 5 of this thesis we will use an optimization routine which operates on the surface of a hyper-sphere. A hyper-sphere is a type of Stiefel manifold which is a well understood geometry and for which optimization routines have been developed. Specifically, we will use a conjugate gradient algorithm on the Stiefel manifold as presented in Edelman et. al. [10]. The chief impediments to adapting an optimization algorithm to a non-Euclidean surface are that the inner product metric on the manifold is not the standard inner product and that line searches and parallel transport must be made along the surface as opposed to straight lines as in Euclidean space. The line searches on the Stiefel manifold will instead be computed as searches along a geodesic. Because the tangent space changes as a vector moves along a geodesic, the parallel transport of a vector must be treated so as to take this into account.

The Stiefel manifold is defined as the set $V_{n,p}$ of all $n \times p$ orthonormal matrices, $\mathbf{Y}$, where $n > p$. That is the set of tall thin matrices which satisfy

$$\mathbf{Y}^T\mathbf{Y} = \mathbf{I}. \tag{2.20}$$

The remainder of this section is divided into three subsections. In the first we will examine the inner product metric and the gradient on the Stiefel manifold. In the second section we will present the geodesic equation and the parallel transport equation for a vector along the geodesic. In the third section we will present the conjugate gradient algorithm on the Stiefel manifold as it will be used in Chapter 5.

2.4.1 Inner Products and Gradients

The inner product between two matrices in Euclidean space is given by

$$\langle \mathbf{\Delta}_1, \mathbf{\Delta}_2 \rangle = \operatorname{tr} \mathbf{\Delta}_1^T \mathbf{\Delta}_2, \tag{2.21}$$

where here $\mathbf{\Delta}_i$ are n-by-p matrices. However, the use of this inner product to define a metric on the Stiefel manifold would be incorrect. The vectors on the Stiefel manifold are not free in all n-by-p dimensions, but are instead constrained in the directions off of the manifold. Consequently, the Euclidean metric would weigh these dimensions twice what they should be weighed. The inner product should properly discount these dimensions which are normal to the surface where the inner product is being calculated. The inner product on the Stiefel manifold is then a function of where on the manifold the inner product is taken and is given by

$$\langle \mathbf{\Delta}_1, \mathbf{\Delta}_2 \rangle_{\mathbf{Y}} = \operatorname{tr} \mathbf{\Delta}_1^T (\mathbf{I} - \frac{1}{2}\mathbf{Y}\mathbf{Y}^T) \mathbf{\Delta}_2, \tag{2.22}$$

where $\mathbf{\Delta}_i$ are n-by-p matrices and $\mathbf{Y}$ is the point on the Stiefel manifold at which the inner product is taken.

The gradient in Euclidean space is taken as the matrix of derivatives with respect to each element. For a function $f(\mathbf{x})$ in Euclidean space, the elements of the gradient are then,

$$[\nabla f(\mathbf{x})]_i = \frac{\partial f(\mathbf{x})}{\partial x_i}. \tag{2.23}$$

In Euclidean space, this matrix would be in the tangent space of the surface and would thus be a proper gradient. On the Stiefel manifold, this matrix could instead be in a direction which is not in the tangent space of the manifold at that point. The gradient must then be corrected by subtracting the element of the gradient which lies in the normal space, this will leave a vector which lies in the tangent space. The gradient on the Stiefel manifold must is computed as

$$\nabla f(\mathbf{Y}) = f_{\mathbf{Y}}(\mathbf{Y}) - \mathbf{Y} f_{\mathbf{Y}}^T(\mathbf{Y})\mathbf{Y}. \tag{2.24}$$

where $f_{\mathbf{Y}}$ is the matrix of derivatives of f with respect to the elements of $\mathbf{Y}$. The second term is the projection of $f_{\mathbf{Y}}$ into the tangent space. With this subtracted, the resultant gradient $\nabla f(\mathbf{Y})$ lies in the tangent space.

2.4.2 Geodesics and Parallel Transport

It will be important to our optimization routine to perform searches along the surface. In a flat space, we would use the equation for a line starting at point $\mathbf{x}_0$ and proceeding in direction $\mathbf{H}$ as

$$\mathbf{x}(u) = \mathbf{x}_0 + u\mathbf{H}. \tag{2.25}$$

On the Stiefel manifold, the line search will be replaced by a search along a geodesic. The geodesic is the shortest path between two points on the manifold which remains on the manifold. The equation for a geodesic starting at the point $\mathbf{Y}_0$ and going in direction $\mathbf{H}$ is given by the equation

$$\mathbf{Y}(u) = \mathbf{Y}_0\mathbf{M}(u) + \mathbf{Q}\mathbf{N}(u). \tag{2.26}$$

where

$$\mathbf{Q}\mathbf{R} = (\mathbf{I} - \mathbf{Y}\mathbf{Y}^T)\mathbf{H} \tag{2.27}$$

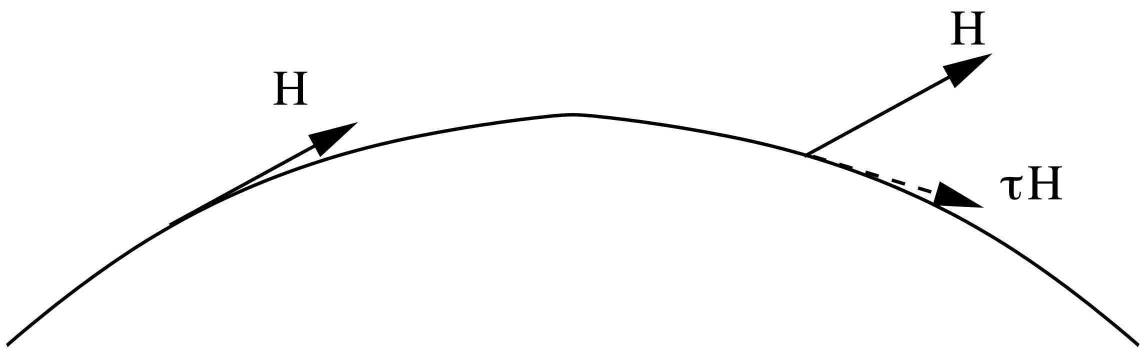


Figure 2.1: Parallel transport of a tangent vector. The vector **H** is to be transported to $\tau\mathbf{H}$ by moving and rotating. The new vector $\tau\mathbf{H}$ remains a tangent vector [10].

is the compact QR-decomposition, and $\mathbf{M}(u)$ and $\mathbf{N}(u)$ are p-by-p matrices given by the matrix exponential

$$\begin{pmatrix} \mathbf{M}(u) \\ \mathbf{N}(u) \end{pmatrix} = \exp u \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_p \\ 0 \end{pmatrix}, \tag{2.28}$$

and $\mathbf{A} = \mathbf{Y}^T\mathbf{H}$.

For the parallel transport of vectors on the Stiefel manifold, we must move the vector to the new location of the manifold and also rotate the vector so that it continues to point in the proper direction along the geodesic. The operation of parallel transport is illustrated in Figure 2.1. The tangent vector **H** on the left is to be transported to its new location on the right. If the vector is simply moved to the new location, we see that it points off of the surface and is no longer a tangent vector as it does not point along the geodesic any longer. To accomplish the transport and have the result be a tangent vector, it must be rotated as it moves such that it continues remains in the tangent space. We will designate this operation by τ and the transport of a vector **H** as $\tau\mathbf{H}$. This is accomplished with an equation similar to the geodesic equation,

$$\tau\mathbf{H} = \mathbf{H}\mathbf{M}(u) - \mathbf{Y}\mathbf{R}^T\mathbf{N}(u). \tag{2.29}$$

2.4.3 Conjugate Gradient on a Stiefel Manifold

The conjugate gradient (CG) algorithm in Euclidean space seeks to minimize a function by a series of line searches. A CG step can be describes as follows: with the starting point $\mathbf{x}_{k-1}$ find the minimum along a line search in direction $\mathbf{H}_{k-1}$ and label this point $\mathbf{x}_k$, then compute the new search direction from the former search direction and the new gradient.

The initial point $\mathbf{x}_0$ of the algorithm must be chosen and the initial search direction is set to the negative of the gradient $\mathbf{G}_k = \nabla f(\mathbf{x}_k)$, as

$$\mathbf{H}_0 = -\mathbf{G}_0. \tag{2.30}$$

The steps of the CG algorithm for minimizing a function $f(x)$ then are:

$$\begin{aligned}
&\text{Line search} && u_{\min} = \arg\min_u f(\mathbf{x}_{k-1} + u\mathbf{H}_{k-1}) && (2.31)\\
&\text{Vector update} && \mathbf{x}_k = \mathbf{x}_{k-1} + u_{\min}\mathbf{H}_{k-1} && (2.32)\\
&\text{Gradient} && \mathbf{G}_k = \nabla f(\mathbf{x}_k) && (2.33)\\
&\text{Conjugate weight} && \gamma_k = \frac{\langle \mathbf{G}_k - \mathbf{G}_{k-1}, \mathbf{G}_k\rangle}{\langle \mathbf{G}_{k-1}, \mathbf{G}_{k-1}\rangle} && (2.34)\\
&\text{New search direction} && \mathbf{H}_k = \mathbf{G}_k + \gamma_k\mathbf{H}_{k-1} && (2.35)
\end{aligned}$$

The new search direction is computed from the old search direction and the present gradient. The weight is determined such as to set the new search direction conjugate to the old search direction, through the Hessian. That is

$$\mathbf{H}_{k-1}^T f_{xx}\mathbf{H}_k = 0. \tag{2.36}$$

The CG algorithm on a Stiefel manifold will replicate these steps with the considerations expressed in the proceeding subsections. Let us consider the minimization of a function $f(\mathbf{Y})$ on a Stiefel manifold. We will of course initialize our algorithm as was done above with a starting point $\mathbf{Y}_0$ and compute the initial search direction as the negative of the gradient $\mathbf{H}_0 = -\nabla f(\mathbf{Y}_0)$, but here the gradient must be computed using the gradient equation (2.24)

We next consider the line search in (2.31). This must become a search along a geodesic using (2.26). After computing $u_{\min}$ we again use the geodesic equation to compute the update vector in (2.32). The gradient in (2.33) is computed again by (2.24). For the last two equations in the algorithm, the conjugate weight and the new search direction, we must consider the parallel transport of the vectors when we make these computations. In computing the conjugate weight, we cannot perform the subtraction in the numerator of (2.34) without first transporting the old gradient to the new $\mathbf{Y}$ since the new gradient is taken at this point. The parallel transport of $\mathbf{G}_{k-1}$ or $\tau\mathbf{G}_{k-1}$ is given by (2.29). Likewise in (2.35) we cannot add the old search direction $\mathbf{H}_{k-1}$ to the new gradient without transporting it in the same manner to produce $\tau\mathbf{H}_{k-1}$. The conjugate gradient algorithm on the surface of the Stiefel manifold is now given as:

$$\text{Geodesic search} \quad \left\{ \begin{aligned} u_{\min} &= \arg\min_u f(\mathbf{Y}(u)) \text{ where} \\ \mathbf{Y}(u) &= \mathbf{Y}_{k-1}\mathbf{M}(u) + \mathbf{Q}\mathbf{N}(u) \\ \mathbf{QR} &= (\mathbf{I} - \mathbf{Y}\mathbf{Y}^T)\mathbf{H}_{k-1} \\ \begin{pmatrix} \mathbf{M}(u) \\ \mathbf{N}(u) \end{pmatrix} &= \exp u \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_p \\ 0 \end{pmatrix} \end{aligned} \right. \tag{2.37}$$

$$\text{Vector update} \quad \mathbf{Y}_k = \mathbf{Y}(u_{\min}) \tag{2.38}$$

$$\text{Gradient} \quad \mathbf{G}_k = \nabla f(\mathbf{Y}_k) = f_{\mathbf{Y}} - \mathbf{Y}_k f_{\mathbf{Y}}^T \mathbf{Y}_k \tag{2.39}$$

$$\text{Conjugate weight} \quad \left\{ \begin{aligned} \tau\mathbf{G}_{k-1} &= \mathbf{G}_{k-1} \text{ or } 0 \\ \gamma_k &= \frac{\langle \mathbf{G}_k - \tau\mathbf{G}_{k-1}, \mathbf{G}_k \rangle_{\mathbf{Y}_k}}{\langle \mathbf{G}_{k-1}, \mathbf{G}_{k-1} \rangle_{\mathbf{Y}_k}} \end{aligned} \right. \tag{2.40}$$

$$\text{New search direction} \quad \left\{ \begin{aligned} \tau\mathbf{H}_{k-1} &= \mathbf{H}_{k-1}\mathbf{M}(u_{\min}) - \mathbf{Y}_{k-1}\mathbf{R}^T\mathbf{N}(u_{\min}) \\ \mathbf{H}_k &= \mathbf{G}_k + \gamma_k \tau\mathbf{H}_{k-1} \end{aligned} \right. \tag{2.41}$$

2.5 Summary

In this chapter we conducted a review of the relevant literature for this thesis. The problems being investigated in this thesis are estimation of the geometric parameters of a target and the classification of the target. In the first part of this chapter, we reviewed several methods for performing these tasks and discussed the strengths and weaknesses. The methods important for the remainder of this thesis are the Fourier Mellin Matched Filter method, the Template Matching method for parameter estimation.

In the second part of this chapter, we reviewed the necessary techniques which will be employed in the remainder of the thesis to solve these problems. We discussed optimization methods which will be necessary for minimization of the cost function discussed in Chapter 1. We put additional emphasis on two algorithms, the Newton Method and the Conjugate Gradient Method. The Newton Method will be used in the parameter estimation work, and the Conjugate Gradient algorithm on a Stiefel manifold which will be used in the classification work.

Chapter 3

Parameter Estimation

In this chapter we will present the multiscale template matching parameter estimation technique. As was discussed in Chapter 1, the problem which we are attempting to solve is the maximum likelihood solution of the template matching problem where size, rotation and location are all unknown. The solution to this problem is complicated by the ill-behavior of the likelihood surface. As was shown in Chapter 1, optimization methods are not sufficient to find the maximum likelihood solution due to the numerous local minima of the cost surface. We will present here an algorithm which attempts to overcome the problems associated with optimizing the likelihood surface by minimizing on an approximate surface generated from an approximate template. By smoothing the true template with a diffusion-like equation, we will develop an iterative template matching algorithm which achieves this optimization.

We will first simplify the problem of likelihood maximization by decoupling the estimation of the location parameters from the estimation of the geometric parameters. This decoupling will be done by applying a heuristic approach to the solution of the location parameters. The advantage here is that the location parameters can be solved in closed form for a given set of geometric parameters by using a Linear Least Squares Estimator (LLSE) [1]. Solving for two of the parameters in closed form greatly reduces the complexity of the problem. Further, by removing consideration of the location parameters from the problem, we can focus exclusively on the problems caused by the geometric parameters when we examine the multiscale template technique. The approach taken for finding the location parameters is a Linear Least Squares Estimator of a delta function [1]. While this is not the maximum likelihood estimator, it has been shown to work well and requires considerably less computation than the ML estimate. After decoupling, the likelihood surface is solely a function of the two geometric parameters, size and rotation. It is with respect to these parameters that we demonstrated the ill-behavior in the likelihood surface with Figure 1.1. This ill-behavior of the likelihood surface will be dealt with by a multiscale approach. We will avoid the ill-behaved optimization problem and will instead estimate on a well-behaved approximation to the likelihood surface by using a smooth template where we have removed those qualities which generate the ill-behavior.

The remainder of this chapter is organized as follows. First, Section 3.1 will demonstrate how the location parameters are decoupled from the geometric parameter estimation using a heuristic approach that is not maximum likelihood but has been shown to work well in practice. In Section 3.2, we will present the work of Abu-Naser et. al. which is used for the estimation of the location parameters. Section 3.3 will examine the conditions upon the smooth approximate templates and will develop a diffusion-like equation for generating a progression of templates which is solvable in the Fourier domain. In Section 3.4 we will bring these two estimations together and present the iterative steps of the overall algorithm. Section 3.5 presents the Newton algorithm for minimizing the approximate cost surfaces. Section 3.6 discusses the ideas behind $t-$schedule selection. Finally Section 3.7 presents the Cramer-Rao bounds upon the estimation of the geometric parameters and the performance criteria which will be used for evaluation.

3.1 Decoupling the Location Estimation

In this section we will demonstrate that the estimation of the location parameters can be decoupled from the estimation of the geometric parameters while the solution remains an ML estimation. As was presented in Chapter 1, we will consider our data as a parameterized template in additive noise as

$$g(\mathbf{r}) = f_E(\mathbf{r} - \bar{\mathbf{r}}^0; \boldsymbol{\theta}^0) + n(\mathbf{r}), \tag{3.1}$$

where the true, unknown parameters of interest are $\bar{\mathbf{r}}^0$ and $\boldsymbol{\theta}^0$.

The cost function associated with the maximum likelihood solution of both the geometric parameters and the location parameters is derived from the Gaussian density. As shown in Chapter 1, it is

$$J(\bar{\mathbf{r}}, \boldsymbol{\theta}|g) = \frac{1}{2\sigma_n^2} ||f_E(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta}) - g(\mathbf{r})||_2^2. \tag{3.2}$$

The maximum likelihood solution for the full parameter set $\{\bar{\mathbf{r}}, \boldsymbol{\theta}\}$ in Gaussian noise is equivalent to the minimization of the cost function J. So if we define the maximum likelihood estimates as $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$, the solution can be expressed as

$$\{\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}}\} = \arg \min_{\{\bar{\mathbf{r}}, \boldsymbol{\theta}\}} J(\bar{\mathbf{r}}, \boldsymbol{\theta}|g). \tag{3.3}$$

Let us consider only a solution for the geometric parameters. This is done by breaking the minimization into two minimizations. The estimate of the geometric parameters as

$$\hat{\boldsymbol{\theta}} = \arg \min_{\{\boldsymbol{\theta}, \bar{\mathbf{r}}\}} J(\bar{\mathbf{r}}, \boldsymbol{\theta}|g) = \arg \min_{\boldsymbol{\theta}} \left(\min_{\bar{\mathbf{r}}} J(\bar{\mathbf{r}}, \boldsymbol{\theta}|g) \right). \tag{3.4}$$

If we now define a solution for minimizing the cost function, J, across $\bar{\mathbf{r}}$ in (3.4) for a fixed set of geometric parameters as

$$\hat{\mathbf{r}}(\boldsymbol{\theta}) = \arg\min_{\bar{\mathbf{r}}} J(\hat{\mathbf{r}}, \boldsymbol{\theta}|g), \tag{3.5}$$

then the substitution of (3.5) into (3.4) yields

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} J(\hat{\mathbf{r}}(\boldsymbol{\theta}), \boldsymbol{\theta}|g). \tag{3.6}$$

Thus, if we now have a solution to (3.5) then we can embed this solution into (3.6) and still achieve a maximum likelihood solution to the full problem. We instead use an approximation of this solution by using a delta reconstruction method. The delta reconstruction method has been shown in practice to work will and can be evaluated in closed form so is more computationally efficient. In the next section we will present the object localization method of Abu-Naser et. al., which we will embed in our maximum likelihood approach to the estimation of the geometric parameters. This approach allows a closed form solution for the location parameters in a two step algorithm for the solution to the joint estimation problem of the location and geometric parameters.

3.2 MAP Estimation of Location

For the location estimation, or the solution to (3.5), we follow the impulse reconstruction technique of Abu-Naser et. al. [1]. In this work, the authors demonstrated template matching with known geometric parameters using an image reconstruction algorithm. The approach in this work, is to take the original image as a delta function at the true target location, as the convolutional equation

$$g(\mathbf{r}) = f_E(\mathbf{r}; \boldsymbol{\theta}^0) * \delta(\mathbf{r} - \bar{\mathbf{r}}^0) + n(\mathbf{r}), \tag{3.7}$$

where $*$ represents linear convolution of the two functions and is defined as

$$f(\mathbf{r}) * g(\mathbf{r}) = \int f(\mathbf{r}') * g(\mathbf{r} - \mathbf{r}') d\mathbf{r}'. \tag{3.8}$$

The data is then considered to be the delta function blurred by a convolutional filter which is the target template with additive noise. Instead of directly estimating the location parameters $\bar{\mathbf{r}}$, we estimate the δ function as if it were the desired image [1]. Proper image reconstruction should produce a distinct peak at the location of the delta function in (3.7). This location is the estimate of $\bar{\mathbf{r}}$ which is the target location. This approach allows greater accuracy with respect to localization and can incorporate background statistics. The incorporation of background statistics produces a solution which is more robust in the presence of noise than the matched filter techniques described earlier.

Many algorithms have been developed for recovering the original image in the presence of blurring and noise. The most straight forward solution is of course the inverse filter of the convolution given above. In most practical situations the inverse filter is high pass and therefore produces noisy reconstructions which are not useful. To counteract the high pass nature of these filters, regularization techniques have been developed to temper the large response in the higher frequencies. As has been shown, the statistically optimal solution is the Wiener inverse filter which regularizes with the statistics of the noise and the object to be reconstructed. In the work of Abu-Naser et. al. they use this regularized inversion filter. Under the assumption of Gaussian noise, the regularized inversion filter is equivalent to the Maximum *a posterior* (MAP) estimation of the object.

For this section we will be assuming that the geometric parameters are fixed and known. With a given set of geometric parameters $\boldsymbol{\theta}$, we can construct an estimate of position following Abu-Naser [1] by first making a MAP estimate of the delta function [1]. Though we desire a sharp delta function, we do not enforce this constraint, and it should be noted that the "delta-reconstruction" is free in all N^2 pixels in the image. To avoid this solution, let us consider that we are estimating a function, $d(\mathbf{r})$, which should, approximately, be a delta-function.

The MAP estimation is then achieved with the filter equation shown here in the Fourier domain as

$$\hat{d}(\mathbf{r}) = \mathcal{F}^{-1}\left\{ \frac{\tilde{f}_E^*(\mathbf{k};\boldsymbol{\theta})}{|\tilde{f}_E(\mathbf{k};\boldsymbol{\theta})|^2 + \left(\frac{\sigma_n}{\sigma_d}\right)^2} \tilde{g}(\mathbf{k}) \right\}, \tag{3.9}$$

where the regularization is imposed by the second term in the denominator which is a ratio of the noise power σ_n^2 to the signal power σ_d^2. In [1], these statistics are unknown, and the estimation is achieved using an expectation maximization algorithm to estimate the signal and noise statistics alongside the delta reconstruction. In this work we have assumed the noise statistics are known and we thus bypass the EM algorithm by using the single closed form solution given above.

The estimate for the target location $\bar{\mathbf{r}}$, or $\hat{\mathbf{r}}(\boldsymbol{\theta})$ is obtained by selecting the point of maximum response of $\hat{d}(\mathbf{r})$ as our position estimate, as

$$\hat{\mathbf{r}}(\boldsymbol{\theta}) = \arg\max_{\mathbf{r}} \hat{d}(\mathbf{r}). \tag{3.10}$$

Substituting (3.10) into (3.6) and expanding J accordingly, we have the maximum likelihood estimator, $\hat{\boldsymbol{\theta}}$ as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \frac{1}{\sigma_n^2} \| f_E(\mathbf{r} - \hat{\mathbf{r}}(\boldsymbol{\theta}); \boldsymbol{\theta}) - g(\mathbf{r}) \|_2^2 \equiv \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; f), \tag{3.11}$$

where we have eliminated the $\bar{\mathbf{r}}$ dependence. The relation (3.11) is now dependent solely upon the geometric parameters $\hat{\boldsymbol{\theta}}$, but still produces an ill-behaved surface. We will now replaced the exact template definition f_E with an approximate template which will induce better behavior in the likelihood surface. In the next section we will examine the smoothed approximate templates.

3.3 Template Progression

To produce a surface which is easily optimizable, we require a model template which has certain properties. Since multimodal templates produce multiple minima on the cost surface, we desire an initial template which is monomodal. Also, discontinuities on the template produce discontinuities on the cost surface, so we also desire that the template be continuous. The first condition will be addressed by having all the templates degrade to a monomodal blob at the lowest resolution. The second condition is addressed by the choice of the equation for the template progression.

Let us generate a library of templates, indexed by t as f_t, ranging from a smooth monomodal template f_0 to the full-detail template $f_\infty = \lim_{t\to\infty} f_t$. The full-detail template is here designated as f_∞ and not as f_E since we cannot assume that the template in the library exactly matches the true template. In most practical applications differences will exist and these must be considered as model mismatch. Further, in the classification section the full-detail template will be composed as a combination of templates in a template library and not as a single exact template, and thus we will have deliberately introduced model mismatch into the template. The template designation f_t now refers to a template at a certain smoothness scale designated by the index t.

A relatively simple choice for the smoothest template, and the one which will be used throughout this work, is a Gaussian blob matched (in the two-norm sense) in size and amplitude to the full-detail template given as

$$f_0(\mathbf{r}) = A \exp\left(-\frac{\|\mathbf{r}\|_2^2}{2w^2}\right). \tag{3.12}$$

The smoothest template will generate a well-behaved surface which is easily optimizable. As detail is added, however, the surface will become less well behaved.

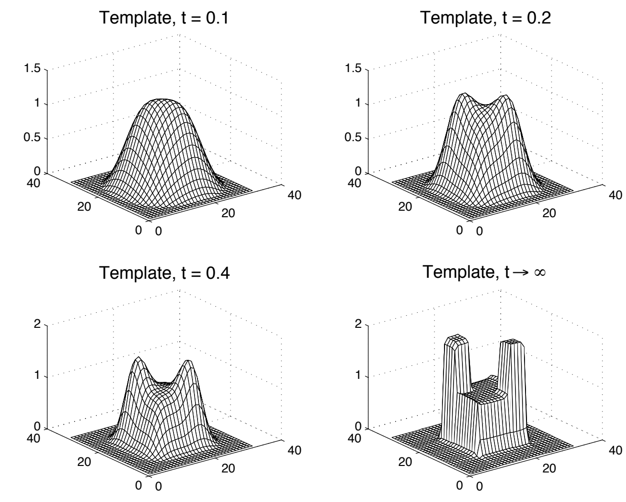
While successive estimation will hopefully locate us inside a "well of attraction" of the global minimum where we will avoid local minima, discontinuities in any template can produce discontinuities anywhere on our surface. We therefore choose a method for generating templates which in the continuous domain produces continuous templates. We choose a diffusion like process which guarantees that for any finite value of t the template will be continuous. We specify the multiscale template f_t in the Fourier domain as

$$\tilde{f}_t(\mathbf{k}) = \left(\tilde{f}_\infty(\mathbf{k}) - \tilde{f}_0(\mathbf{k})\right) \exp\left(\frac{-\|\mathbf{k}\|^2}{t}\right) + \tilde{f}_0(\mathbf{k}), \tag{3.13}$$

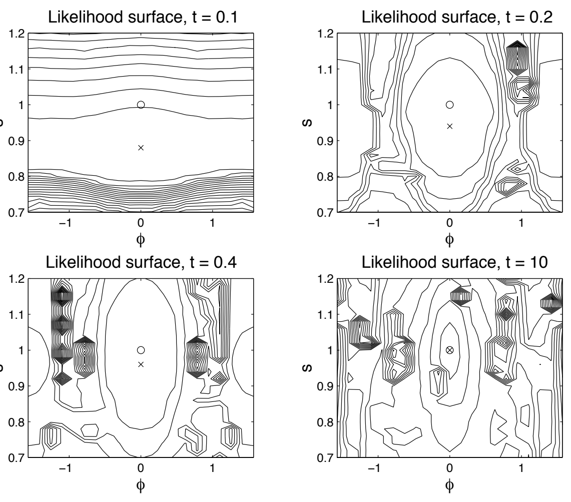
where $\mathbf{k}$ is the spatial frequancy variable.

A progression through the template library for the two peaked template is shown in Figure 3.1(a). At small values of t, the template is a smooth approximation of the true template; it is essentially a Gaussian blob at the lowest value of t. As t is increased, the template begins to sharpen, until as t approaches infinity the true template emerges. Associated with this are the likelihood surfaces which are related the template at each value of t. Here we see that at smaller values of t, the surface is very smooth, has no rotational localization, and has a very broad scale localization. It is obvious that the rotational estimation has little value and the scale estimation only marginal value since the minimum of this surface is so broad. However, minimization of this surface is simple, and the resultant minimization is generally in the area of the global minimum of the surface produced using the full-detail template.

As t increases, the ill-behavior which was seen in the exact likelihood surface returns, but previous estimates have placed us within the area of the global minimum. We can then add detail to the template and initiate a minimization where the last minimization settled, which is hopefully within a well-behaved "well" on the likelihood surface. This estimate will then become more accurate than the last. We can continue this procedure as we continue to add detail to the template. The final solution for the



(a)



(b)

Figure 3.1: In (a) we see four templates computed using the diffusion-like equation (3.13) for four different values of t. In (b) we see contour plots of the likelihood surfaces associated with each template. We see that the surface becomes more ill-behaved as we add detail to the template, but the minimum of each surface from an optimization routine (noted by the x) becomes progressively closer to the global minimum (noted by o). By starting the new minimization by the previous estimate, we remain within a well-behaved basin around the global minimum.

parameters will of course be a local minimum, but may not necessarily be the global minimum. We will see later that the rate at which the templates evolve influences whether the final solution will be the global minimum with slower t-schedules leading to more accurate estimates.

3.4 Algorithm Description

The procedure which the estimation algorithm will follow was outlined in the previous section. In this section, we will detail the individual steps of the algorithm as they will be implemented throughout this work. As was outlined above, the algorithm will produce an approximate template from the diffusion-like equation. Then estimates will be produced successively for the location parameter and the geometric parameters. Detail will then be added to the template and the procedure continued.

The specific steps of the algorithm are:

1. Begin at $t = 0$. This starts with the smoothest template and best behaved likelihood surface. The initial template is given by the best match (in the 2-norm) Gaussian function, given by (3.12).

2. Construct f_t for the current value of t with (3.13).

3. Compute the ML estimate via the equations

$$\hat{\mathbf{r}}_t(\boldsymbol{\theta}) = \arg\max_{\mathbf{r}} \mathcal{F}^{-1}\left\{ \frac{\tilde{f}_t^*(\mathbf{k};\boldsymbol{\theta})}{|\tilde{f}_t(\mathbf{k};\boldsymbol{\theta})|^2 + \left(\frac{\sigma_n}{\sigma_\delta}\right)^2} \tilde{g}(\mathbf{k}) \right\}. \tag{3.14}$$

$$\hat{\boldsymbol{\theta}}_t = \arg\min_{\boldsymbol{\theta}} \frac{1}{\sigma_n^2} \| f_t(\mathbf{r};\boldsymbol{\theta}) * \delta(\mathbf{r} - \hat{\mathbf{r}}_t(\boldsymbol{\theta})) - g(\mathbf{r}) \|_2^2. \tag{3.15}$$

   The minimization in 3.15 is performed via a Newton algorithm given in Section 3.5.

4. Raise t and proceed to step 2. The t-schedule should be chosen to take small steps at low values of t for which the information in the estimate is changing quickly and then larger steps for higher values. This agrees with the method for graduated non-convexity [30]. Our method is to choose a small value for the first t and to double it for each subsequent t. If a longer schedule is desired, a smaller multiplier is used. The t-schedule can be terminated when the information content of the template is essentially maximized, or the template contains sufficient detail.

3.5 Newton Step Description

The minimization mentioned which was mentioned in the previous section is done using a Newton algorithm [31]. In this section we will develop the necessary pieces of the Newton iteration. The Newton optimization algorithm implemented here seeks to minimize the squared error between the image produced from the template at the estimated target location and geometric parameters and the data. The error can be calculated as

$$\hat{\boldsymbol{\theta}}_t = \arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}; f_t) = \arg\min_{\boldsymbol{\theta}} \int e_t^2(\mathbf{r}; \boldsymbol{\theta})\, d\mathbf{r} \tag{3.16}$$

where e_t is the error image given as

$$e_t(\mathbf{r}; \boldsymbol{\theta}) = f_t(\mathbf{r}; \boldsymbol{\theta}) * \delta(\mathbf{r} - \hat{\mathbf{r}}_t(\boldsymbol{\theta})) - g(\mathbf{r}). \tag{3.17}$$

The procedure which the Newton iteration follows is to produce updates of the parameter set vector as [31]

$$\begin{aligned} (\mathbf{U}_t(\boldsymbol{\theta}^{(i)}) + \mathbf{S}_t(\boldsymbol{\theta}^{(i)}))\mathbf{p}^{(i)} &= -\int \mathbf{J}_t(\mathbf{r}; \boldsymbol{\theta}^{(i)}) e_t(\mathbf{r}; \boldsymbol{\theta}^{(i)}) d\mathbf{r} && (3.18) \\ \boldsymbol{\theta}^{(i+1)} &= \boldsymbol{\theta}^{(i)} + \mathbf{p}^{(i)}. && (3.19) \end{aligned}$$

Here, $\mathbf{p}^{(i)}$ is the update vector for the parameter set, $\mathbf{J}_t$ is the Jacobian vector of the error image e_t, and $\mathbf{U}_t$ and $\mathbf{S}_t$ are functions of the Jacobian and Hessian as described below.

The Jacobian vector is calculated as the vector of first derivative functions of the of the error image as

$$\begin{aligned}
\mathbf{J}_t(\mathbf{r};\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} e_t(\mathbf{r};\boldsymbol{\theta}) = \left[\frac{\partial e_t(\mathbf{r};\boldsymbol{\theta})}{\partial s} \quad \frac{\partial e_t(\mathbf{r};\boldsymbol{\theta})}{\partial \phi}\right]^T \\
&= \nabla_{\boldsymbol{\theta}} f_t(\mathbf{r};\boldsymbol{\theta}) * \delta(\mathbf{r}-\hat{\mathbf{r}}_t(\boldsymbol{\theta})) + f_t(\mathbf{r};\boldsymbol{\theta}) * \nabla_{\boldsymbol{\theta}} \delta(\mathbf{r}-\hat{\mathbf{r}}_t(\boldsymbol{\theta})) \qquad (3.20) \\
&= \nabla_{\boldsymbol{\theta}} f_t(\mathbf{r}-\hat{\mathbf{r}}(\boldsymbol{\theta});\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta}) \nabla_{\mathbf{r}} f_t(\mathbf{r}-\hat{\mathbf{r}}(\boldsymbol{\theta});\boldsymbol{\theta}). \qquad (3.21)
\end{aligned}$$

3.5.1 Approximation of the Jacobian and the Hessian

The first term of (3.21) is easily computable from the template function, however the second term is troublesome. The gradient $\nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta})$ is not guaranteed to exist everywhere, and even if it does exist, it cannot be calculated in closed form since $\hat{\mathbf{r}}(\boldsymbol{\theta})$ involves a maximization and also the function $\hat{\mathbf{r}}(\boldsymbol{\theta})$ is not actually ever a continuous function in implementation since it exists on the pixilated image. Numerical estimation of the gradient $\nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta})$ is complicated by the large granularity of the image pixels with respect to the usual size of $\nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta})$. By this, we mean that if we were to attempt to approximate the value of an element of $\nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta})$ (for example $\frac{\partial \hat{r}_x(\boldsymbol{\theta})}{\partial s}$) by the approximate derivative relation

$$\frac{\partial \hat{r}_x(s,\phi)}{\partial s} \approx \frac{\hat{r}_x(s+\Delta s,\phi) - \hat{r}_x(s,\phi)}{\Delta s}, \qquad (3.22)$$

we find that the numerator is either zero (if the two estimates of position are coincident on the same pixel), or one or more pixels in distance, thus making the ratio in (3.22) arbitrarily large (if they are different pixels) because Δs can be made arbitrarily small. We can instead attempt an alternative method to get a more accurate approximation of the elements of $\nabla_{\boldsymbol{\theta}} \hat{\mathbf{r}}(\boldsymbol{\theta})$. First we raise s, from zero, incrementally until $\hat{r}_x(s,\phi)$

$\lvert\nabla_{\boldsymbol{\theta}} f(\mathbf{r};\boldsymbol{\theta})\rvert$	$\frac{\partial \hat{r}_x}{\partial s}$		$\frac{\partial \hat{r}_y}{\partial s}$		$\frac{\partial \hat{r}_x}{\partial \phi}$		$\frac{\partial \hat{r}_y}{\partial \phi}$	
mean	mean	std	mean	std	mean	std	mean	std
1.3534	0.0090	0.0334	0.0090	0.0334	0.0087	0.0406	0.0087	0.0406

Table 3.1: Values of elements of $\nabla_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta})$ for 1095 iterations. It is seen that the values of $\nabla_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta})$ are small relative to $\lvert\nabla_{\boldsymbol{\theta}} f(\mathbf{r};\boldsymbol{\theta})\rvert$.

produces at least a one pixel difference and let this point be s_1. We then find a similar point by lowering s and designating this point to be s_2. We can then calculate the gradient element approximation as

$$\frac{\partial \hat{r}_x(s,\phi)}{\partial s} \approx \frac{\hat{r}_x(s_2,\phi) - \hat{r}_x(s_1,\phi)}{s_2 - s_1}. \tag{3.23}$$

The disadvantage with this method is that it is computationally intensive since we must calculate $\hat{\mathbf{r}}(\boldsymbol{\theta})$ at many values of s to get an accurate solution to (3.23). However, by using this method on representative data and templates, we find that the elements of the gradient matrix of the location estimation $\nabla_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta})$ are typically two to three orders of magnitude below the gradient matrix of the template $\nabla_{\boldsymbol{\theta}} f_t(\mathbf{r}-\hat{\mathbf{r}}(\boldsymbol{\theta});\boldsymbol{\theta})$, i.e. the other term of (3.21). A sample of the values for these matrices are shown in Table 3.1.

Therefore, it will not significantly decrease the accuracy of the Jacobian $\mathbf{J}_t(\mathbf{r};\boldsymbol{\theta})$ if we disregard this term. By eliminating the second term, we can then compute an approximation for the Jacobian as

$$\mathbf{J}_t(\mathbf{r};\boldsymbol{\theta}) \approx \nabla_{\boldsymbol{\theta}} f_t(\mathbf{r}-\hat{\mathbf{r}}(\hat{\boldsymbol{\theta}});\boldsymbol{\theta}). \tag{3.24}$$

The second term which is also important for the calculation of the Newton step is the Hessian of the error image, or the matrix of second derivatives. The Hessian of the error image is given as

$$\begin{aligned}\mathbf{T}_t(\mathbf{r};\boldsymbol{\theta}) = \nabla^2_{\boldsymbol{\theta}} e_t(\mathbf{r};\boldsymbol{\theta}) = \big[\nabla^2_{\boldsymbol{\theta}} f_t(\mathbf{r};\boldsymbol{\theta}) - \nabla^2_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta})\nabla_{\mathbf{r}} f_t(\mathbf{r};\boldsymbol{\theta}) \\ -2\nabla_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}}\nabla_{\mathbf{r}} f_t(\mathbf{r};\boldsymbol{\theta}) + (\nabla_{\boldsymbol{\theta}}\hat{\mathbf{r}}(\boldsymbol{\theta}))^2\nabla^2_{\mathbf{r}} f_t(\mathbf{r};\boldsymbol{\theta})\big] * \delta(\mathbf{r} - \hat{\mathbf{r}}(\boldsymbol{\theta}))\end{aligned} \tag{3.25}$$

Similar to the problems associated with calculation of the Jacobian, the terms of the Hessian which have the gradients of $\hat{\mathbf{r}}(\boldsymbol{\theta})$ can be computationally troublesome. With a similar procedure as that shown for the Jacobian we can demonstrate that these terms are usually not significant, so we will disregard these terms in the calculation of the approximate Hessian. This leaves, as an approximate Hessian, the expression

$$\mathbf{T}_t(\mathbf{r};\boldsymbol{\theta}) \approx \nabla^2_{\boldsymbol{\theta}} f_t(\mathbf{r} - \hat{\mathbf{r}}(\boldsymbol{\theta});\boldsymbol{\theta}). \tag{3.26}$$

3.5.2 Calculation of the Newton Step

The two matrices on the left hand side of (3.18) are computed from the Jacobian and Hessian. We will actually use the approximate relations which were presented in the previous section. The first matrix of (3.18) is the inner product of the Jacobian with itself, and the second matrix of (3.18) is the integral of the Hessian components with the error image. Thus, we will calculate the Jacobian inner product as

$$\mathbf{U}_t(\boldsymbol{\theta}) = \int \mathbf{J}_t(\mathbf{r};\boldsymbol{\theta})\mathbf{J}_t^T(\mathbf{r};\boldsymbol{\theta})\, d\mathbf{r}. \tag{3.27}$$

For the second matrix, we need to integrate the Hessian components with the error image. Specifying the elements of $\mathbf{S}_t(\boldsymbol{\theta})$, by $s_{i,j}(\boldsymbol{\theta})$ and the elements of $\mathbf{T}_t(\mathbf{r};\boldsymbol{\theta})$ by $t_{i,j}(\mathbf{r};\boldsymbol{\theta})$, we calculate the matrix as

$$s_{i,j}(\boldsymbol{\theta}) = \int t_{i,j}(\mathbf{r};\boldsymbol{\theta}) e_t(\mathbf{r};\boldsymbol{\theta})\, d\mathbf{r}. \tag{3.28}$$

The Newton algorithm iteration is then done by calculating (3.18) and (3.19).

These equations are iterated until the value of the likelihood as evaluated in (3.15) ceases to change significantly. Expressed in algorithmic terms, the iteration continues until,

$$J(\boldsymbol{\theta}_k; f_t) - J(\boldsymbol{\theta}_{k-1}; f_t) < \tau \tag{3.29}$$

for some small value of τ which is the termination criteria.

3.6 Computing the t-Schedule

The last item of significance which we need before implementation of the algorithm is to define how the t-schedule is determined. The t-schedule determines the values of t for which the Newton iteration algorithm is run. As explained earlier, the values of t must be chosen to induce better behavior in the likelihood surface for the minimization by the Newton algorithm. The selection of the appropriate values of t has a direct affect upon the amount of computation and whether the algorithm will converge to a local minimum or the global minimum. The better behavior is induced by smoothing, or flattening, the surface about the global minimum. It seems appropriate then that we wish to base the values of t upon the expected value of the local curvature of the likelihood surface around the global minimum.

The expected local curvature of the likelihood surface at the true parameters for any value of t can be calculated by the expected values of the elements of the Hessian matrix [20]. The elements of the Hessian matrix, with respect to the parameters, is given as

$$E\frac{\partial^2 l_t}{\partial\theta_i\partial\theta_j} = \frac{2}{\sigma_n^2}\int (f_t(\mathbf{r};\boldsymbol{\theta}) - f_E(\mathbf{r}))\frac{\partial^2 f_t(\mathbf{r};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i\partial\boldsymbol{\theta}_j}\mathbf{r} + \frac{2}{\sigma_n^2}\int \frac{\partial f_t(\mathbf{r};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_i}\frac{\partial f_t(\mathbf{r};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}_j}\mathbf{r}. \tag{3.30}$$

In Figure 3.2 we plot the value of the local curvature with respect to t for the two-peaked target examined earlier. We see that the value of the curvature is small for small value of t when the template is a smooth approximate template. The curvature

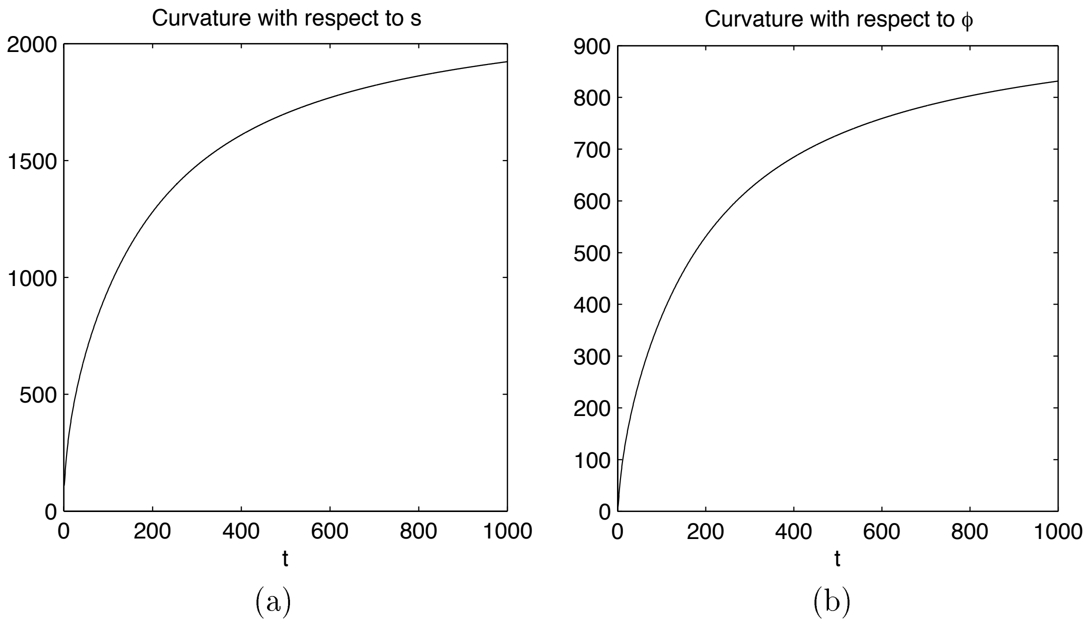


Figure 3.2: The expected local curvature around the global minimum of the cost surface for the two-peaked template example for (a) the scale parameter and (b) the rotation parameter.

quickly increases as t is raised until it asymptotically approaches the curvature of the exact template. For efficient and accurate estimation, it is important that we step through t quickly, however, if t increases too rapidly then we are likely to end up in a local minimum. At $t = 0$, the template is smoothest and the surface is the most well behaved and has the broadest well around the global minimum. The template must evolve in such a way that the estimate remains within the well. Using the curvature as a gauge of this well, we see that initially small changes in t are necessary so as not to too drastically change the surface. However as t increases larger steps can be taken since the surface evolves slower with respect to t. In this thesis, we will use a geometric progression for the values of t. That is, we will begin at some value and raise this value by a multiplicative amount repeatedly until the algorithms finishes. This type of schedule is similar to that used in the Graduated Non-Convexity approach in [4, 29, 30].

3.7 Performance and Bounds

Closely related to the curvature of the surface is the Cramer-Rao bound on the variance of the parameter estimation. The CRB for the estimates is arrived at by inverting the Fisher information matrix, which is computed by evaluating the Hessian for the exact template at the true geometric and location parameters. The CRB establishes the lower limit on the variance of the estimates for an unbiased estimator. The CRB is only achievable for an unbiased estimator for a parameter for which a sufficient statistic exists. If such a statistic is not available, then the CRB cannot be achieved. Nonetheless, it is still useful to examine the lower limit to the variance of the estimates as it is often possible to approach the performance limit of the CRB even if it is unachievable..

As was mentioned above, the CRB is calculated from the inverse of the Fisher information matrix. The Fisher information matrix can be calculated from the second derivatives of the log-likelihood equations as

$$\mathbf{I} = \begin{bmatrix} E\left(\frac{\partial^2 l}{\partial s^2}\right) & E\frac{\partial^2 l}{\partial s \partial \phi} & E\frac{\partial^2 l}{\partial s \partial r_x} & E\frac{\partial^2 l}{\partial s \partial r_y} \\ E\frac{\partial^2 l}{\partial s \partial \phi} & E\left(\frac{\partial^2 l}{\partial \phi^2}\right) & E\frac{\partial^2 l}{\partial \phi \partial r_x} & E\frac{\partial^2 l}{\partial \phi \partial r_y} \\ E\frac{\partial^2 l}{\partial s \partial r_x} & E\frac{\partial^2 l}{\partial \phi \partial r_x} & E\left(\frac{\partial^2 l}{\partial r_x^2}\right) & E\frac{\partial^2 l}{\partial r_x \partial r_y} \\ E\frac{\partial^2 l}{\partial s \partial r_y} & E\frac{\partial^2 l}{\partial \phi \partial r_y} & E\frac{\partial^2 l}{\partial r_x \partial r_y} & E\left(\frac{\partial^2 l}{\partial r_y^2}\right) \end{bmatrix}. \tag{3.31}$$

Then from the elements of the inverse of the information matrix we can compute the lower bounds upon the variance of the parameter estimations, as

$$\text{var s} \geq [\mathbf{I}^{-1}]_{1,1} \tag{3.32}$$

$$\text{var } \phi \geq [\mathbf{I}^{-1}]_{2,2} \tag{3.33}$$

$$\text{var } \mathbf{r} \geq [\mathbf{I}^{-1}]_{3,3} + [\mathbf{I}^{-1}]_{4,4}. \tag{3.34}$$

In Figure 3.3 we have computed the CRB of the parameters estimations versus

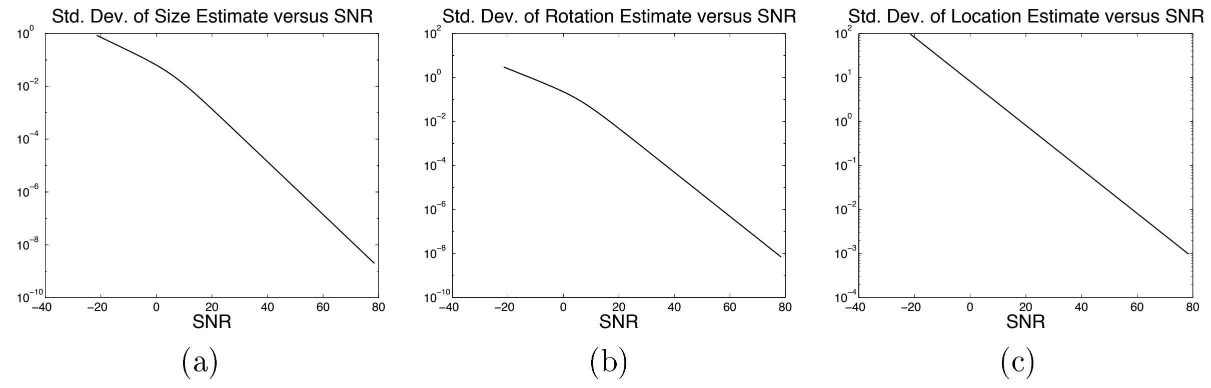


Figure 3.3: The Cramer-Rao bounds for estimation of the parameters versus SNR. In (a) the scale parameter, (b) the rotation parameter and (c) the location parameter.

noise. These behave as expected with exact estimation possible in the no noise case and estimation performance degrading as noise increases.

The other importance of the Cramer-Rao bounds for this work is that we will use the bounds to define a good and a bad estimation. We consider that the true global minimum of the cost surface can vary from the expected position. The amount of this variation is determined by the Cramer-Rao bound equations shown above. We therefore can assume that the final estimation of the parameters was close to the global minimum of the cost surface if the estimated parameters are within an ellipse computed as a small multiple of the Cramer-Rao bound from the true parameters. The axes of the ellipse are computed from (3.32) and (3.33). Any final estimation which falls within the ellipse can be designated a “hit”. Estimations outside the ellipse are assumed invalid and designated “misses”. The multiple of the CRB which we will use is 3 times the Cramer-Rao bound.

3.8 Summary

In this chapter, we have developed an algorithm for estimation of the geometric parameters of a target given a functional template. The primary focus of the estimation technique presented is to overcome the ill-behavior of the likelihood surface which is due to the complexity of the template. The method to alleviate the ill-behavior is to use an approximate template which was presented in Section 3.3. The approximate template is computed using a diffusion-like equation to compute a cross between the exact template and an ideal approximate template. In Section 3.4, we presented the steps of the algorithm which will use this template in an iterative estimation routine where detail is slowly added to the templates and the estimation is progressively refined until the full detail template is reached and the most accurate estimation is achieved. We continued this investigation in Section 3.5 by calculating the steps of the Newton algorithm which is used for the minimization of the likelihood equation at each iteration. Section 3.6 presented the method for computing the t-schedule, or the progression of templates to be used in the minimization. Lastly Section 3.7 presented the performance criteria which will be used to define a successful or unsuccessful estimation.

In the next Chapter, we will examine the performance of this algorithm for several examples and for Monte Carlo simulations.

Chapter 4

Parameter Estimation Results

In the last chapter we presented an estimation algorithm for simultaneously estimating the location parameters of a target along with the geometric parameters using a multiscale template. In this chapter we will present results of running the algorithm against synthetic data and against infrared and optical examples. The performance metric which we will use is whether the final estimate is sufficiently close to the values of the true parameters. We will define a "hit" as a final estimation which is within an ellipse with axes which are calculated as 3 times the Cramer-Rao bound around the true parameters. A "miss" is any estimation which falls outside this range. The ellipse is defined to approximate the area within which we would expect the global minimum to fall. We will show examples which demonstrate that the algorithm is able to compute the true parameters for a variety of templates. Additionally, through Monte Carlo simulation we will show that the algorithm performs well at a wide range of SNR and geometric parameters.

The remainder of this chapter is organized as follow. In Section 4.1, we will present Monte Carlo results of running the algorithm versus synthetic data for several levels of SNR and for two t-schedules. We will demonstrate here that the number of hits can be increased by a slower t-schedule which reduces the frequency of getting caught in local minima. In section 4.2, we will present examples of the algorithm's performance versus infrared and optical data and we will compare the results to the performance of the FMMF. These data are representative of real estimation problems with non-white Gaussian noise and background clutter. Last in section 4.3 we examine the range of parameters over which the algorithm correctly converges to the true parameters. Here we show that the algorithm is fairly robust to the starting conditions and accurately converges over a wide range of true parameter values for both size and rotation.

4.1 Monte Carlo Performance

In order to demonstrate the performance of the algorithm we will perform Monte Carlo simulations versus a set of simulated data. We will generate synthetic scenes with the two-peaked template shown earlier and additive Gaussian noise at various levels and run the algorithm with the two-peaked template. The output of the algorithm is the geometric parameters along with the location parameters. We will display the results as both an accumulation of "hits" and "misses" and by showing scatter plots which demonstrate the width of the scattering of the estimation results.

When we plot the final estimated geometric parameters on a scatter plot, we see that the majority of the estimates are clustered around the true parameters while some of the estimates land in other positions. As was explained earlier, to quantify this result, we generate an ellipse defined by the Cramer-Rao bound. The global minimum will vary from the true parameters by some amount on each simulation. While stochastic, the location of this minimum will, with high probability, be within three times the CRB of the true parameters. We can confidently call successful those estimates which lie within this ellipse. Final estimates which land within the ellipse are designated "hits", while those outside the ellipse are designated "misses".

In Figure 4.1, we show scatter plots of estimates from 400 simulations of the two-peak example at varying signal to noise ratios. The true parameters for the target in the Monte Carlo simulations are a relative size of 0.5 and a rotation of 0.25 radians.

Also shown are the CRB ellipses which define accurate estimates. First, it should be noted that the size of the ellipses decreases as SNR increases. This is a result of the lowering of the CRB as SNR increases. With the scatter plots, we observe that in 4.1(a–c), the number of estimates which are outside the ellipse, and are therefore considered misses, decreases as SNR increases. This shows improved performance as SNR increases. However, the number of misses increases for the highest value of SNR due to a bias which appears in the size estimates. This bias when combined with the increased tightness of the CRB causes more misses to be registered at higher values of SNR than may be warranted by the fact that these estimates are close to the true parameters. Further study into the bias phenomenon may be needed to decrease this effect.

Table 4.1 summarizes the Monte Carlo runs of the example just presented and for an additional t-schedule. The additional t-schedule will be used to demonstrate the effect of a slower progression upon the success of the estimations. Each line of the table shows the result of 400 simulations of running the algorithm to estimate the template parameters for a specific level of noise and the indicated t-schedule. The two t-schedules are differentiated by the number of stops employed. The long t-schedule has 8 stops, and the short t-schedule has 4 stops. We should expect the longer schedule to perform better since there will be less chance of getting caught in a local minima due a template progression which is too fast.

The error in the estimates is composed of two components, one caused by the variance of the local minimum around the true parameters, and the second caused by the algorithm becoming trapped in a local minimum which is far from the true parameters. The first type of error is bounded below by the CRB, and in practice is usually so small as to effect the target location estimation by less than one pixel. The second type of error constitutes a catastrophic miss by the algorithm. This error is demonstrated in Figure 4.2 with a pair of Monte Carlo runs at two different t-schedules. From the first scatter plot, we see that the majority of estimates are grouped around

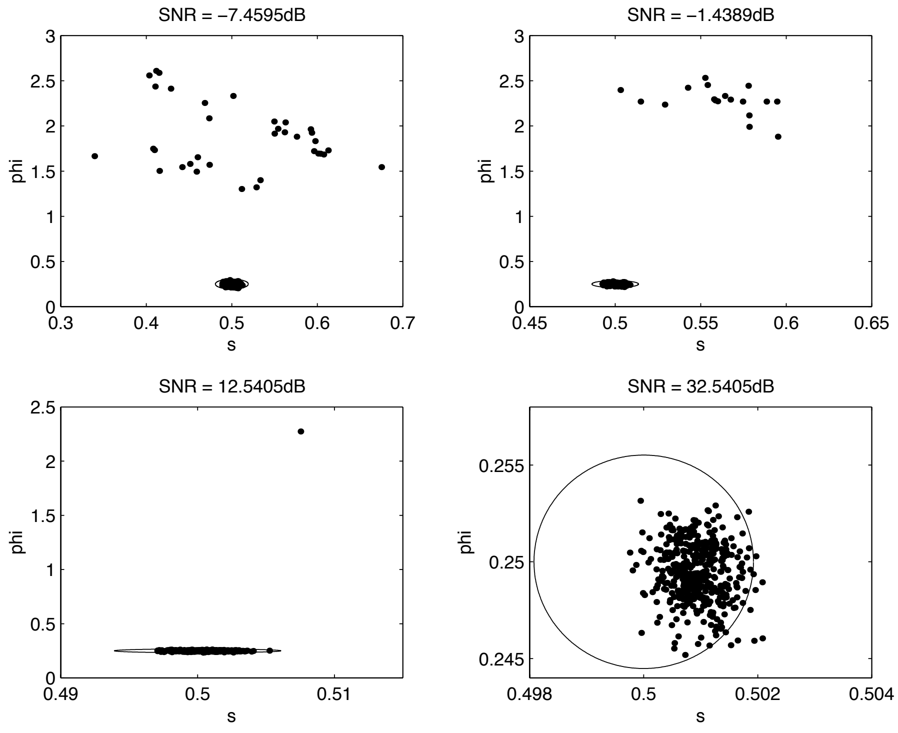


Figure 4.1: Scatter plots of estimations from 400 simulations of the two-peak example. The ellipse is drawn at 3 times the CRB around the exact parameters, and is used to define "good" estimations.

		Size		Rotation		$\hat{r}_x$		$\hat{r}_y$	
SNR (dB)	Schedule	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
-7.4595	short	0.50538	0.020603	0.42739	0.57641	64.8575	3.4557	65.0825	2.5947
-1.4389	short	0.50357	0.013774	0.33951	0.42302	64.9575	2.2921	65.0325	1.5756
12.5405	short	0.50105	0.0025995	0.25451	0.10889	65.025	0.5	64.9825	0.35
18.5611	short	0.50097	0.00095061	0.24918	0.0034543	65	0	65	0
32.5405	short	0.50104	0.00013374	0.24942	0.00046999	65	0	65	0
38.5611	short	0.50105	0.0002985	0.24943	0.0010505	65	0	65	0
-7.4595	long	0.5013	0.024512	0.38909	0.4705	64.785	2.5238	64.905	1.6379
-1.4389	long	0.50221	0.015672	0.32243	0.34788	64.895	1.9425	65.0325	1.1245
12.5405	long	0.50084	0.0014079	0.25412	0.10135	65.03	0.6	64.985	0.3
18.5611	long	0.50085	0.00096719	0.24918	0.0035128	65	0	65	0
32.5405	long	0.50093	0.00043025	0.24944	0.0015144	65	0	65	0
38.5611	long	0.50092	0.00030425	0.24944	0.0010707	65	0	65	0

Table 4.1: Estimation errors for size, rotation and location.

the true parameters, but 35 out of 400 of the estimates missed dramatically. These are instances where the algorithm became caught in a local minimum which was not the global minimum, and thus resulted in an enormous error. The second scatter plot shows the same Monte Carlo runs with a t-schedule which has twice as many stops and thus has twice the computational burden. Here, the number of misses was reduced from 35 to 18. By progressing through the t-schedule at a slower rate, we can reduce the chances of become trapped in a local minimum at the expense of more computations.

4.2 Estimations with Real Data

Figure 4.3 shows an example of running the algorithm to estimate the size, rotation and location of a vehicle in an IR image. Shown first in (a) is the target template which is being used as the functional template. Shown in (b) is the data image. This is an example of a low contrast target in a non-Gaussian cluttered background. Below this, in (c) and (d), are the estimated values for size and rotation parameters for each iteration of the algorithm. The iterations are divided into several sections by vertical dotted lines denoting the respective value of t for each stage of the algorithm. Each stage is denoted by the value of t, which represents a progressively more detailed template being used. We can see that the algorithm converges in 60 iterations to

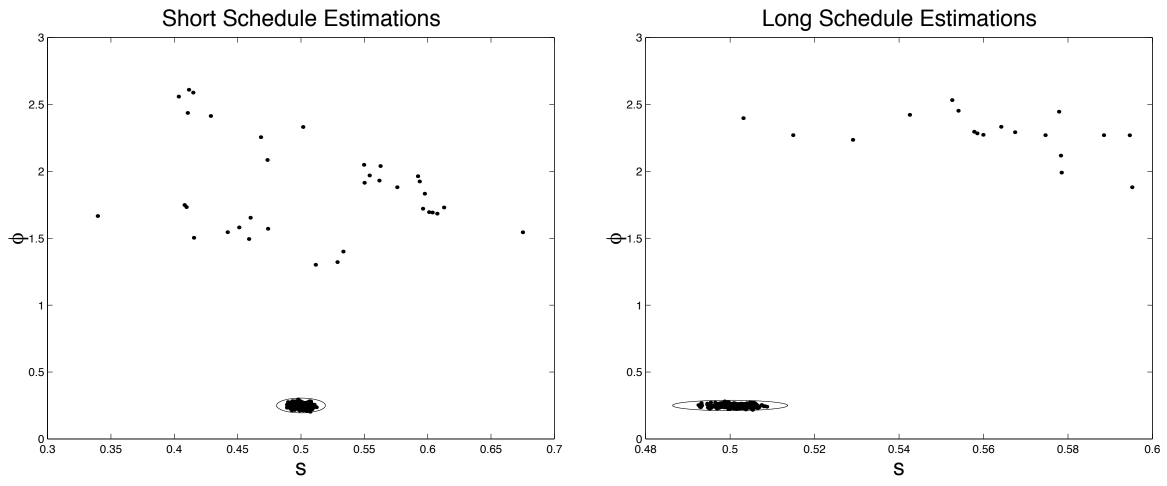


Figure 4.2: Scatter plots of estimations from 400 simulations of the two-peak example for two schedules. Longer schedules result in less misses.

values which appear to match the template to the data appropriately. Also shown in (b) is the final position estimate of the target which also appears to be in the correct vicinity to the center of the target.

The next example, in Figure 4.4, shows an estimation of the geometric parameters for a real optical image using the algorithm. In this case, the template is generated from an image of a cup. The first image in (a) shows the data scene which is being used. The scene is a moderately difficult image since the clutter is of a similar magnitude to the target. The image in (b) shows the template placed at the final estimated position from the algorithm with the appropriate size and rotation as estimated. In (c) and (d), the size and rotation estimates at each iteration are shown. We can see from these plots that the algorithm settled into the final estimates after 42 iterations and that it is an accurate estimate.

4.2.1 Results of the FMMF

In this section, we will make some comparisons between the performance of the current algorithm and the Fourier Mellin Matched Filter (FMMF) [5]. We will present a qualitative example which is the identical data to that presented in the previous

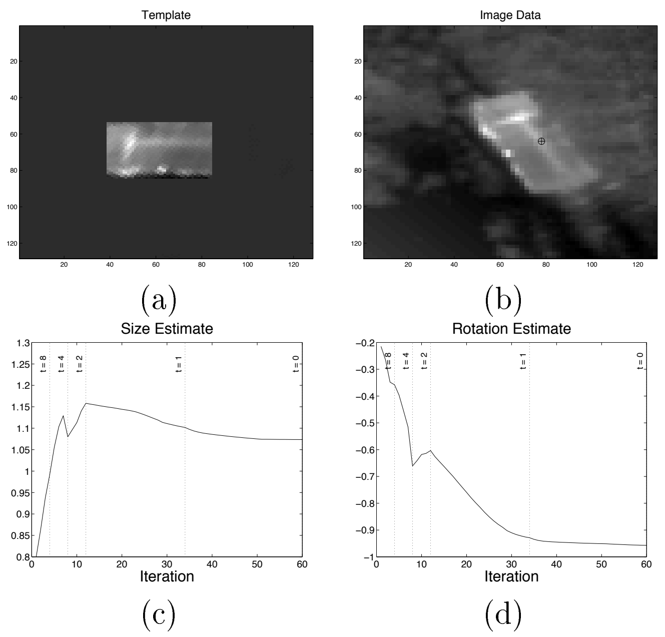


Figure 4.3: Infrared target example. (a) The template (b) the data with estimated center location (c) the estimated size at each iteration (d) the estimated rotation at each iteration.

section of the cup in a cluttered scene and then some Monte Carlo runs to compare the variance of the two estimators.

Figure 4.5 shows FMMF run against the cluttered optical image scene in the previous section in Figure 4.4. In Figure 4.5, we show the output of the Fourier-Mellin matched filter which fails to isolate the true size and rotation for this image. The performance of the FMMF is degraded by the clutter in the scene. A possible source of this degradation is that the FMMF does not operate locally, but across the entire scene. The template matching on the other hand is isolated to the support of the template. If the template were to be located badly then the performance would degrade. This is also a possible rationale for increasing the ability of the algorithm to operate over multiple target scenes.

In Figure 4.6 the performance for the Monte Carlo runs is compared with that of the Fourier Mellin Matched Filter [5]. The plot shows the standard deviation of the error in the location estimate versus SNR. We see that at high SNR, both methods

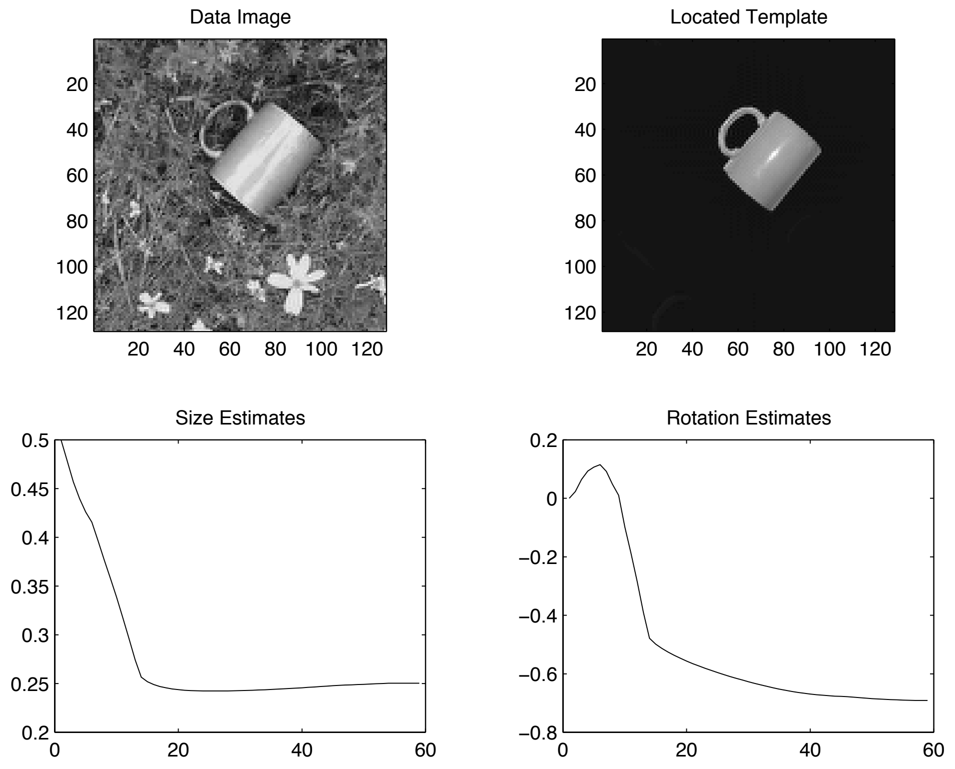


Figure 4.4: Optical example of a cup in a cluttered scene. (a) The data (b) the template at estimated size, rotation and location (c) the estimated size at each iteration (d) the estimated rotation at each iteration.

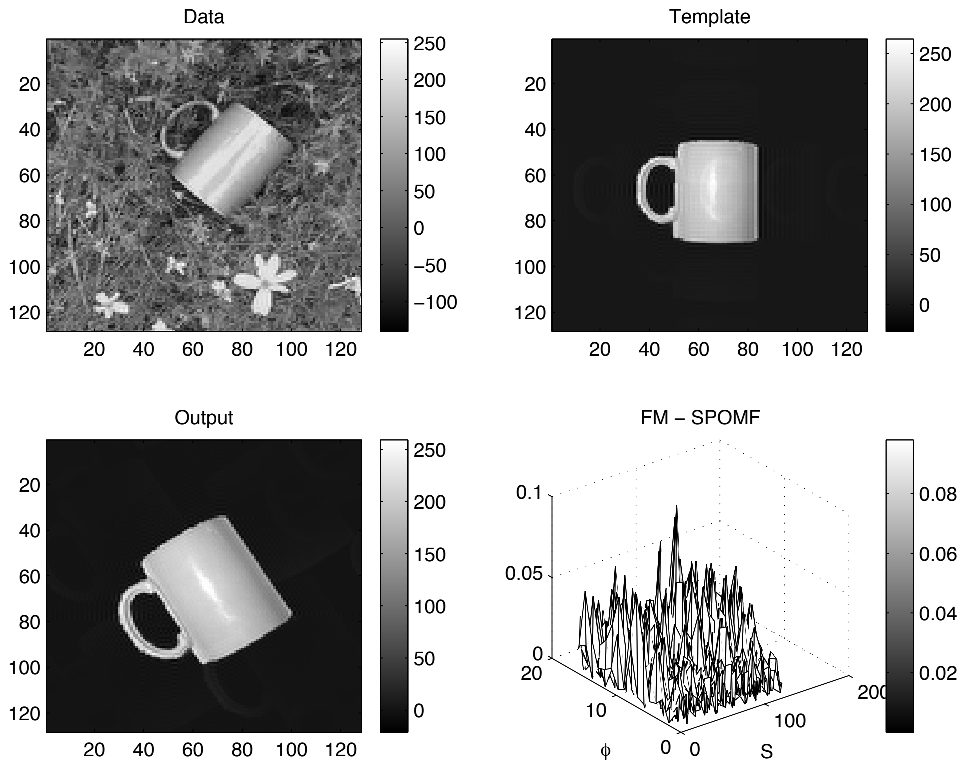


Figure 4.5: The FMMF of the cup in a cluttered scene. The FMMF fails to accurately estimate the parameters.

perform well, but as the SNR decreases then error in the FMMF increases at a faster rate.

4.3 Robustness to Geometric Parameters

The last topic of consideration for the algorithm performance is to determine the ranges of the scale and rotation parameters over which the algorithm converges to the proper values consistently. This would be highly dependent upon the shape of the template, the resolution to which template data is available and the t-schedule used. Thus we can only demonstrate the performance for a specific example. We will show this analysis for the two-peak example which is used throughout this thesis. We will use the short t-schedule which was discussed earlier. Monte Carlo simulations were again run for a variety of targets with values of the scale parameter of $\{.2, .4, .6, .8, 1.0\}$ and values of the rotation parameter of $\{0, \pi/6, \pi/3, \pi/2\}$. The noise variance was set to produce a SNR of 10 dB across the support of the target. The algorithm was initialized at a value of scale of 1.0 and a value of rotation of 0.0 radians. In Table 4.2 we summarize the results of this experiment by itemizing the number of misses out of 20 simulations. The table shows that the algorithm converges to the proper parameters across a wide range of values of the rotation and scale parameters. Specifically, the algorithm converges to the correct parameters consistently for scale values down to .6 with zero rotation. For values of rotation higher than $\pi/3$, the algorithm often converged to a local minimum around $\pi/2$. Thus for low values of rotation the algorithm converges consistently to the correct parameter for a wide range of scale. For correct scale, the algorithm usually converged correctly for most rotations. When both parameters are far distant, convergence is not as consistent.

Interestingly, the performance at higher rotations improves for smaller scale objects. We believe that this is a result of the smoothing operation. Since the smoothing is constant across template size, the relative amount of smoothing is greater for smaller

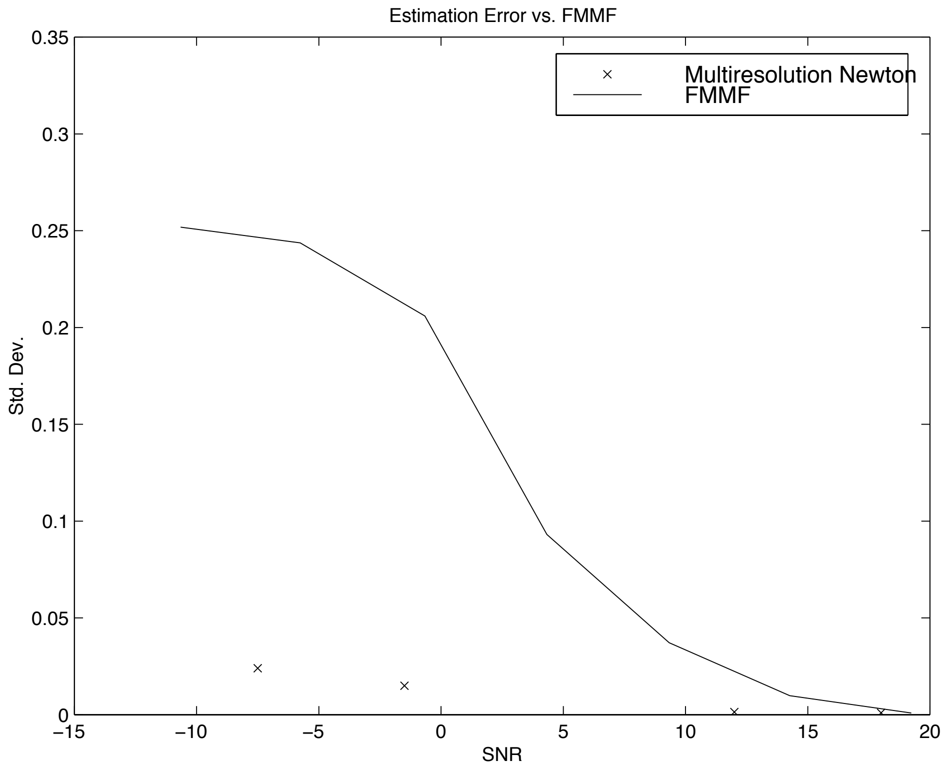


Figure 4.6: Estimation error for location for 400 simulations using the Fourier Mellin Matched Filter and the current algorithm. The current algorithm performs significantly better at low SNR values.

		ϕ^0			
		0^o	30^o	60^o	90^o
	1.0	0	0	15	13
	0.8	0	0	15	14
s^0	0.6	2	2	4	2
	0.4	3	3	2	2
	0.2	8	8	7	6

Table 4.2: The total misses out of 20 Monte Carlo simulations of estimating several scales and rotations. Proper estimation was usually achieved for scales down to .6, and for rotations less than 30^o.

scale objects. This results in an effectively finer t-schedule for small scale parameters than the t-schedule for larger scale objects. We believe that if the t-schedule were made sufficiently fine, performance would improve for all the values in Table 4.2. The problem of optimal t-scheduling would require us to define a cost function for determining the t-schedule which would trade the amount of computation for the final accuracy of the estimation. It would be an interesting topic of future study to examine the problem of optimal t-scheduling.

4.4 Algorithm Performance Summary

In this chapter we examined the performance of the algorithm versus both synthetic and real data examples. For quantitative analysis of the algorithm, we examined Monte Carlo simulations of the algorithm versus the two-peak synthetic target example used throughout this work. We saw that the performance of the algorithm is good versus a wide range of SNR. In addition, we examined the effects of using a longer t-schedule. We saw that the number of misses can be reduced by increasing the number of stops in the t-schedule and thus the amount of computation. Additional stops in the t-schedule allows the estimation to adjust more rapidly with respect to the change in the cost surface. This causes the algorithm to become trapped in local minima less often. Next, we examined the performance of running against real images. We presented two examples, one of a low contrast target in an IR image, and another of an optical image which had significant clutter. Here the algorithm performed well and showed more robust performance than was seen in the FMMF. Lastly, we examined the robustness of the algorithm performance versus a range of geometric parameters. We see that the algorithm performs well across a wide range of geometric parameters.

In the next part of this thesis, we will extend the work on the multiscale estimation algorithm to the problem of target classification. The target classification problem is fundamentally different than that of parameter estimation. Nonetheless, by employing a similar method of progressively adding detail to the target template, we can initialize the algorithm with a general template and delay the classification decision until more information is obtained. This will be accomplished by introducing a template which can be steered between a number of templates in a template library.

Chapter 5

Classification

We continue this work into the area of target classification. As discussed in Chapter 2, target classification is often done by classification of points in a feature space [3, 9, 28, 35–37], where features are computed which are invariant to the unknown geometric parameters and classification of the resultant feature vector is thus invariant. However, in computing features much of the target information is discarded. As an alternative one could compute a score based upon a template match and compare this score with that of other templates in the library. This approach is complicated by the unknown geometric parameters which makes conducting the match difficult and would thus affect the accuracy of the likelihood scores. Here, we present an approach for applying the parameter estimation algorithm which was developed and analyzed in the last two chapters to the problem of classification, essentially solving the joint problem of target classification and parameter estimation.

Under the approach presented in the previous chapters, all targets which are properly normalized in support and amplitude degenerate to an identical Gaussian blob. Therefore, for all these targets, the parameter estimation would begin identically. As estimates are made regarding the scale, rotation, and location parameters, the accuracy of the target classification improves. If we describe the exact template for the template progression routine as a combination of templates from a template library and parameterize this combination with regards to the weight given to each template, then we need only define a "steering vector" which holds these weights. As the algorithm progresses from the smooth approximation towards the exact template, the steering vector is updated as with any of the other parameters and will hopefully be steered towards the correct template. The final values of the steering vector can then be interpreted as a classification with the maximum value designating the selected template. An example of this algorithm is shown in Figure 5.1. The multiscale templates are shown in (a), progressing from a smooth blob to the exact. They are positioned, sized and rotated as estimated. In (b) and (c) we show the convergence of the geometric parameters to close to the true parameters. In (d) the values of steering vector coefficients are shown. The final values are close to being exclusively made of the two-peaked template.

Implementation of this approach requires the development and analysis of several new components to our existing algorithm. First, we need to create a steerable template which is properly parameterized for the template combination. Second, we must create a cost function which is also optimizable with respect to the steering vector. Third, we must consider the minimization of the cost function with respect to the steering vector and incorporate this minimization into the existing algorithm.

The remainder of this chapter is organized as follows: In Section 5.1 we will examine the construction of the steerable template and the steering vector and embed this into the template progression equation. We will define three domains for the steering vector, whether it will exist on a simplex with two separate limiting methods or on a hyper-sphere and we will discuss the relative advantages of both. In Section 5.2 the Stiefel manifold conjugate gradient algorithm as presented in Section 2.2 will be adapted to the problem at hand for the minimization of the cost function with respect to the steering vector. In Section 5.3 we will demonstrate how the coefficients of the steering vector can be related to the posterior probabilities of the templates and thus how they represent a confidence measure on the final template classification. In

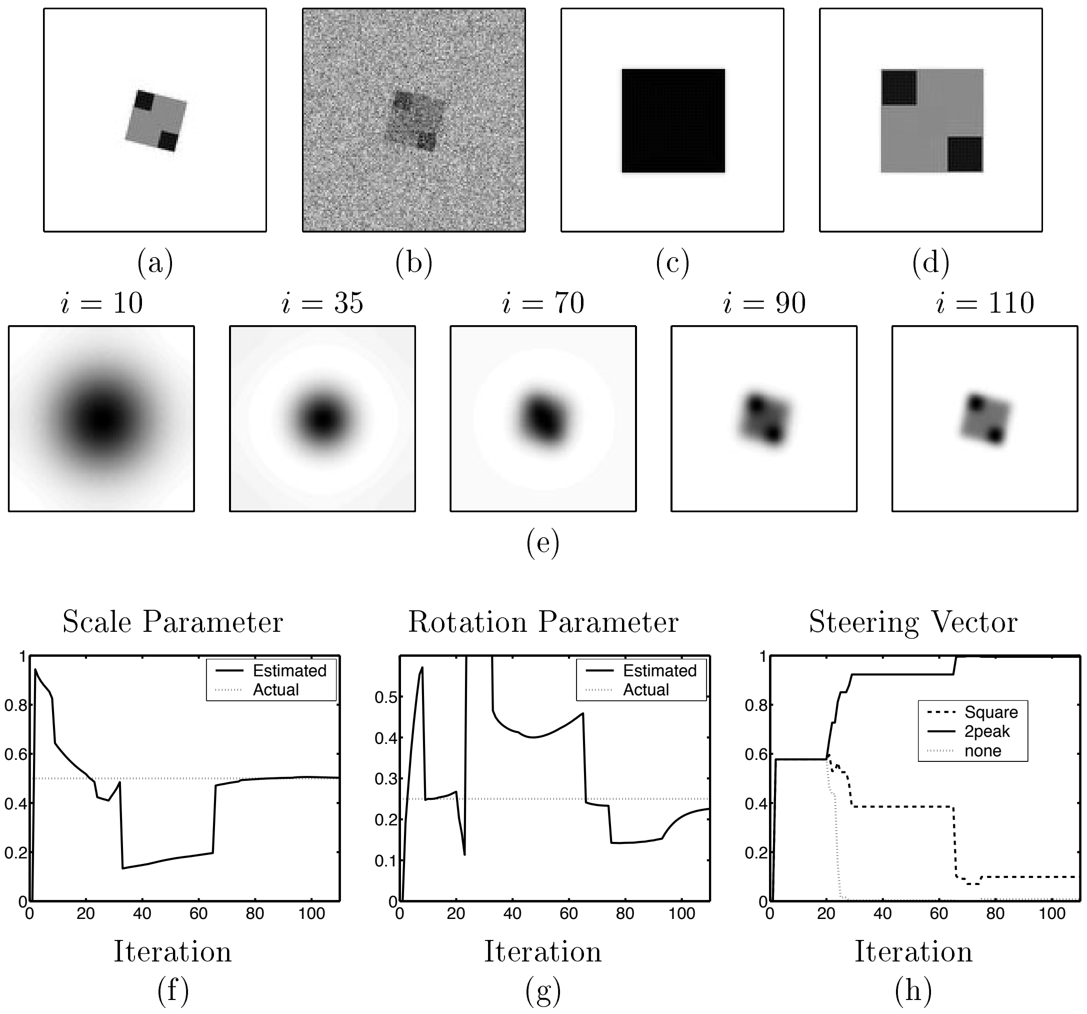


Figure 5.1: In (a) the actual target is shown. In (b) the object is shown in 0 dB SNR noise; this is the data. In (c) and (d), the templates in the template library are shown, there is also a null template. In (e), the estimated template is shown for 5 of the iterations of the algorithm. In (f) the scale parameter is shown with the true parameter. In (g) the rotation parameter is shown with the true parameter. In (h) the values of the steering vector are shown.

Section 5.4 we will introduce two prior model penalty functions which will be used to impose on the solution that it be one of the canonical templates in the library. And lastly we will summarize the several methods presented in Section 5.5.

5.1 Steerable Template

To implement the algorithm for a multiple template library, we must have a method for computing a single template from multiple source templates. The resultant template will be a combination in some way of the several templates in the library and then will be smoothed in the same manner that the single template is smoothed. Therefore, for any smoothness parameter t, we will have a continuously indexed template f_t which is composed from our template library $f^{(k)}$, where k is the index of the templates in the library. For example, our template library may consist of two templates where $f^{(1)}$ is a square, $f^{(2)}$ is a triangle and

$$f_0(\mathbf{r};\boldsymbol{\theta},\boldsymbol{\lambda}) = \lambda_1 f^{(1)}(\mathbf{r};\boldsymbol{\theta}) + \lambda_2 f^{(2)}(\mathbf{r};\boldsymbol{\theta}) \tag{5.1}$$

is the full resolution steerable template with the geometric parameter vector $\boldsymbol{\theta}$ and the steering vector $\boldsymbol{\lambda} = [\lambda_1\ \lambda_2]^T$. The steering vector designates the weight given to each template in the combination of templates. The template $f_0(\mathbf{r};\boldsymbol{\theta},\boldsymbol{\lambda})$ is then the full resolution template which will be used in the diffusion-like equation as was done in the parameter estimation algorithm.

5.1.1 Steerable Templates on a Simplex

We can now define the how the combination of templates which constructs the "steerable template" is computed. An obvious choice here would be a convex linear combination of templates, which would be expressed as

$$f_0(\mathbf{r};\boldsymbol{\theta},\boldsymbol{\lambda}) = \sum_k \lambda_k f^{(k)} \qquad \text{where} \qquad \sum_k \lambda_k = 1 \qquad \text{and} \qquad 0 < \lambda_k < 1, \quad (5.2)$$

where the weight elements λ_i make up the steering vector $\boldsymbol{\lambda}$. This leads to a search space for the steering vector which is a simplex in $K-1$ dimensions, where K is the number of templates in the template library. In this formulation of the steerable template, we have a boundary constraint since the values of λ_k must be constrained to be between 0 and 1. Also the sum constraint must also be imposed.

We consider two methods to impose the simplex boundary. The first is to implement a hard bound constraint and use a constrained optimization algorithm. With this method, line searches are terminated at the boundaries. The second method is to include a highly restrictive cost for points which exist off of the simplex. The cost function then becomes

$$J(\bar{\mathbf{r}},\boldsymbol{\theta},\boldsymbol{\lambda};g(\mathbf{r})) = \frac{1}{2\sigma_n^2}\|g(\mathbf{r}) - f_0(\mathbf{r}-\bar{\mathbf{r}};\boldsymbol{\theta},\boldsymbol{\lambda})\|^2 + \Omega(\boldsymbol{\lambda}), \quad (5.3)$$

where $\Omega(\boldsymbol{\lambda})$ is the additional cost for values of $\boldsymbol{\lambda}$ which are off of the simplex. In this work, we define this cost term as

$$\Omega(\boldsymbol{\lambda}) = -\alpha\lambda_-^2 \qquad \text{where} \qquad \lambda_- = \sum_{\lambda_i<0} \lambda_i. \quad (5.4)$$

For a large value of α, this imposes a restrictive barrier on values of $\boldsymbol{\lambda}$ which lie off of the simplex. As the steering vector approaches the simplex from outside, the value of $\Omega(\boldsymbol{\lambda})$ goes to zero. Squaring the value of λ_- ensures that the first derivative of the cost function is continuous at the boundary.

We will designate the first method (that of hard bound constraints) to be the simplex with boundary conditions, and designate the second method (that of a restrictive cost barrier) to be the simplex with barrier conditions.

5.1.2 Steerable Templates on a Hypersphere

The steerable template formulations given in the previous section have the disadvantage that implementation on the simplex search space will require the implementation of a constrained optimization algorithm to impose both the boundary constraint and the sum constraint. The hard constraints will complicate the running of the algorithm, especially at the early stages when the gradient on this surface will be very small or zero and the constraints would be likely to be imposed often. We would prefer instead that the steering vector exist on a surface which more naturally imposes the necessary constraints. If we instead define the template combination as

$$f_0(\mathbf{r};\boldsymbol{\theta},\boldsymbol{\lambda}) = \sum_k \lambda_k^2 f^{(k)} \qquad \text{where} \qquad \sum_k \lambda_k^2 = 1. \tag{5.5}$$

then the steering vector exists on a K dimensional hyper-sphere. The K dimensional hyper-sphere is a Stiefel manifold of the variety $V_{K,1}$. We now have a template domain space which imposes the unity constraint but which does not have hard boundaries requiring a constrained optimization or cost function imposed barrier. We must now seek an optimization algorithm which remains on this surface. The Stiefel manifold conjugate gradient algorithm described in Section 2.2 will be useful for this problem.

Figure 5.2 shows an example of how the Stiefel manifold would look for the simplest case, which is a two template library. In the two template case, the steering vector is a two element vector where the squared value of each of the elements indicates the relative contribution of the associated library template to the complete template. Thus if the steering vector is at $[1,\ 0]^T$ or $[-1,\ 0]^T$, the complete template is equivalent to the first template in the library $f^{(1)}$. Likewise, the complete template would be

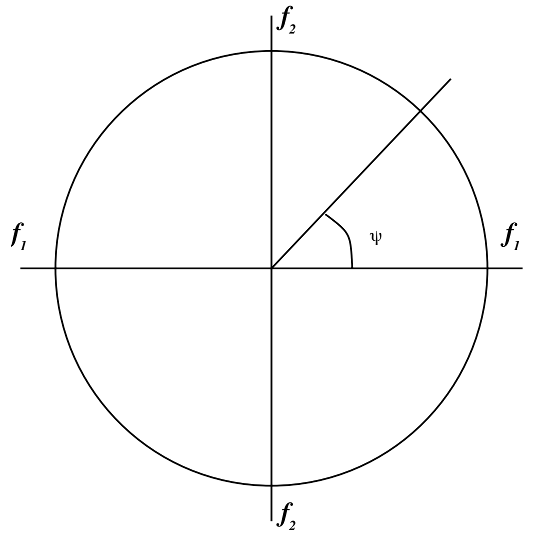


Figure 5.2: The manifold geometry for a two template library. Template one is located at $(1, 0)$ and $(-1, 0)$, while template 2 is located at $(0, 1)$ and $(0, -1)$. The angle ψ is angular rotation from the x-axis.

equivalent to the second template when the steering vector is equal to either $[0,\ 1]^T$ or $[0,\ -1]^T$. Obviously, there is a non-uniqueness of the templates with respect to points on the manifold surface which have elements which are the negative of each other since the template is invariant to the negation of an element. However, this is not a problem for the optimization since, through symmetry, each quadrant of the surface is identical to all the other quadrants.

To express the steerable template more compactly, we will define a matrix $\mathbf{F}(\boldsymbol{\theta})$ which has as its columns the different templates in the template library, as

$$\mathbf{F}(\boldsymbol{\theta}) = \left[\cdots\ f^{(k)}(\mathbf{r};\boldsymbol{\theta})\ \cdots\right]. \tag{5.6}$$

The steerable template can then be expressed using the template matrix as

$$f_0(\mathbf{r};\boldsymbol{\theta},\boldsymbol{\lambda}) = \mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\lambda}\circ\boldsymbol{\lambda}), \tag{5.7}$$

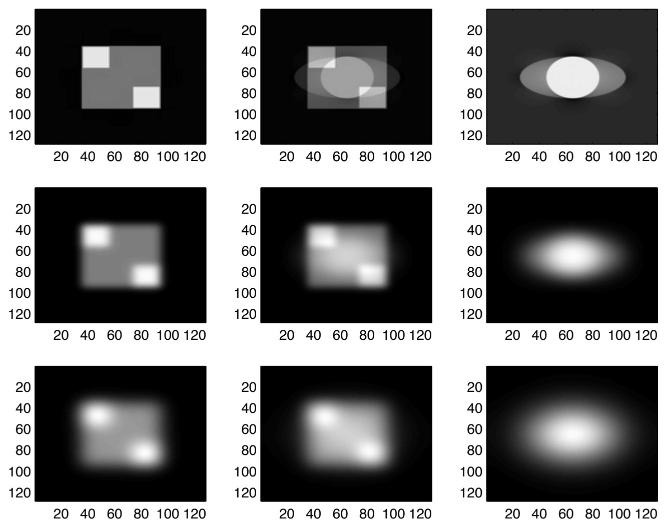


Figure 5.3: An example of a steerable template. The template is entirely composed of the two-peak template on the left and the "eye" template on the right. In the middle, the template is a combination of the two. The smoothing parameter t increases in the downward direction.

where $\circ$ designates the Hadamard product of two vectors or the pointwise multiplication of the elements. We can then insert the steerable template equation into the smoothing equation as given in (3.13) and the full template description becomes

$$\tilde{f}_t(\mathbf{k}) = \left(\tilde{\mathbf{F}}(\mathbf{k})(\boldsymbol{\lambda} \circ \boldsymbol{\lambda}) - \tilde{f}_\infty(\mathbf{k})\right) \exp\left(-\|\mathbf{k}\|^2 t\right) + \tilde{f}_\infty(\mathbf{k}) \tag{5.8}$$

where the unparameterized $\tilde{\mathbf{F}}$ is the normalized Fourier transform of the template library.

In Figure 5.3 we show an example of the steerable template for a two template library. Moving across the images from left to right, the steering vector changes from being exclusively template $f^{(1)}$ to being exclusively template $f^{(2)}$. The center column shows a combination of the two templates and represents an intermediate value of the steering vector. Vertically, we show how the smoothing acts upon the templates, erasing any differences as smoothing increases.

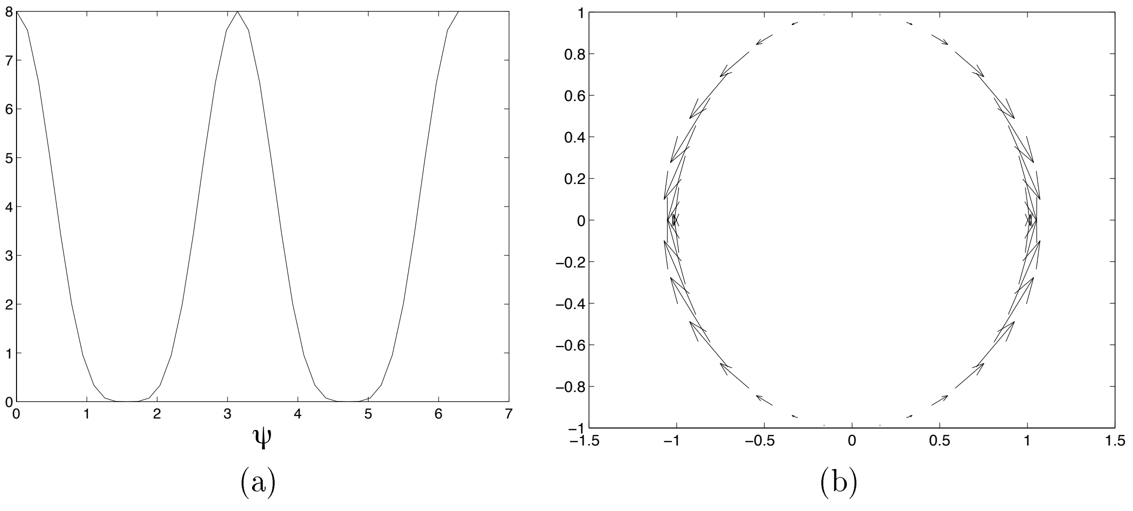


Figure 5.4: (a) shows the cost as a function of angular rotation while (b) shows the gradient vectors on the manifold.

We can now express the cost function for the multiscaled steerable template as

$$J(\bar{\mathbf{r}}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r})) \stackrel{\triangle}{=} \frac{1}{2\sigma_n^2} \|g(\mathbf{r}) - f_t(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta}, \boldsymbol{\lambda})\|^2. \tag{5.9}$$

The cost function $J(\bar{\mathbf{r}}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r}))$ is now a function of the continuously valued vector $\boldsymbol{\lambda}$ instead of the discrete parameter k. It is also parameterized by the smoothing parameter t and the geometric parameters $\boldsymbol{\theta}$ as was the cost function in the previous chapters.

In Figure 5.4 we show how the cost function would appear for two templates when the geometric parameters and the location are known. The cost function is shown as a function of the angular rotation from the positive x-axis. The template surface is represented in Figure 5.3. The angular value of template $f^{(1)}$ is then at 0 and π and the angle for template $f^{(2)}$ is $\frac{\pi}{2}$ and $\frac{3\pi}{2}$. With respect to the cost function in Figure 5.4(a), we see that the value is minimized for the correct template which is $f^{(2)}$, at an angular rotations of $\frac{\pi}{2}$ or $\frac{3\pi}{2}$. The cost function increases as the steering vector moves to templates which include more of $f^{(1)}$.

The only difficulty remaining with the minimization of the new cost function is providing an algorithm which can properly remain on the Stiefel surface while searching for the minimum of the cost function $J(\bar{\mathbf{r}}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r}))$. As was mentioned previously, we will use the Conjugate Gradient algorithm described in Section 2.2 to minimize the cost function on the Stiefel manifold.

5.2 Conjugate Gradient for the Steering Vector

To adapt the conjugate gradient (CG) algorithm presented in Section 2.2 to our problem, we will examine each of the steps of the algorithm individually and compose the requisite equations. The cost function which we are minimizing is $J(\bar{\mathbf{r}}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r}))$ across the steering vector $\boldsymbol{\lambda}$. We will keep all other parameters fixed throughout the minimization so for simplification of the notation we may regard the cost function being minimized as $J(\boldsymbol{\lambda})$. We will initialize the algorithm with a starting point $\boldsymbol{\lambda}_0$ and an initial search direction, which is the negative of the gradient of $J(\boldsymbol{\lambda})$, $\mathbf{H}_0$. We can then proceed with the iterative algorithm.

As was presented in Section 2.2, the CG algorithm is composed of five steps. The first step is a search along a geodesic for the minimum value of the cost function. The geodesic is given by equations (2.37). The construction of the geodesic begins with a QR decomposition

$$\mathbf{QR} = (\mathbf{I} - \boldsymbol{\lambda}\boldsymbol{\lambda}^T)\mathbf{H}_{p-1} \tag{5.10}$$

$$= \mathbf{H}_{p-1} - \boldsymbol{\lambda}\boldsymbol{\lambda}^T\mathbf{H}_{p-1}. \tag{5.11}$$

If we recognize that the search direction $\mathbf{H}_{p-1}$ is tangent to the surface and that the vector $\boldsymbol{\lambda}$ is normal to it, then the second term of (5.11) is zero by orthogonality and the QR decomposition shown is simply a decomposition of the gradient $\mathbf{H}_{p-1}$. This simplifies the QR decomposition to calculating the normalized gradient direction and the magnitude of the gradient as

$$\mathbf{R} = ||\mathbf{H}_{p-1}|| \tag{5.12}$$

$$\mathbf{Q} = \frac{1}{||\mathbf{H}_{p-1}||}\mathbf{H}_{p-1}. \tag{5.13}$$

The next equation for constructing the geodesic is the computation of $\mathbf{M}(u)$ and $\mathbf{N}(u)$, which will be scalar valued for this problem since the manifold is $V_{K,1}$. The values of $\mathbf{M}(u)$ and $\mathbf{N}(u)$ are arrived at from the matrix exponential equation,

$$\begin{pmatrix} \mathbf{M}(u) \\ \mathbf{N}(u) \end{pmatrix} = \exp u \begin{pmatrix} \mathbf{A} & -\mathbf{R}^T \\ \mathbf{R} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_p \\ 0 \end{pmatrix}, \tag{5.14}$$

where $\mathbf{A} = \boldsymbol{\lambda}_{p-1}^T \mathbf{H}_{p-1}$, which as we saw earlier is zero since $\boldsymbol{\lambda}_{p-1}$ is orthogonal to $\mathbf{H}_{p-1}$. So the resultant matrix exponential equation is

$$\begin{pmatrix} \mathbf{M}(u) \\ \mathbf{N}(u) \end{pmatrix} = \exp u||\mathbf{H}_{p-1}|| \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{I}_p \\ 0 \end{pmatrix} \tag{5.15}$$

$$= \begin{pmatrix} \cos(||\mathbf{H}_{p-1}||u) \\ \sin(||\mathbf{H}_{p-1}||u) \end{pmatrix}. \tag{5.16}$$

The scalar values of $\mathbf{M}(u)$ and $\mathbf{N}(u)$ reduce to sinusoids. Then, the resultant geodesic equation is

$$\boldsymbol{\lambda}(u) = \boldsymbol{\lambda}_{p-1}\cos(||\mathbf{H}_{p-1}||u) + \frac{\mathbf{H}_{p-1}}{||\mathbf{H}_{p-1}||}\sin(||\mathbf{H}_{p-1}||u). \tag{5.17}$$

The minimization is computed along this geodesic by varying u and computing the value of $J(\boldsymbol{\lambda}(u))$. The result is then designated to be $u_{\min}$. The vector update $\boldsymbol{\lambda}_p$ is then computed from $u_{\min}$ as

$$\boldsymbol{\lambda}_p = \boldsymbol{\lambda}(u_{\min}). \tag{5.18}$$

Next, we compute the an update to the gradient of the cost function on the Stiefel manifold using (2.39) which projects the Euclidean gradient of $J(\boldsymbol{\lambda})$ into the tangent space of the manifold. If we calculate the vector of derivatives for our cost function as,

$$J_{\boldsymbol{\lambda}} = \begin{bmatrix} \vdots \\ \frac{\partial J}{\partial \lambda_i} \\ \vdots \end{bmatrix}, \tag{5.19}$$

then we can use the derivative vector $J_{\boldsymbol{\lambda}}(\boldsymbol{\lambda})$ to calculate the gradient vector ∇J as

$$\nabla J_0 = J_{\boldsymbol{\lambda}_0} - \boldsymbol{\lambda}_0 J_{\lambda_0}^T \boldsymbol{\lambda}_0 \tag{5.20}$$

$$= (-4\mathbf{F}(\boldsymbol{\theta})^T \mathbf{g} + 4\mathbf{F}(\boldsymbol{\theta})^T \mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\lambda}_0 \circ \boldsymbol{\lambda})) \circ \boldsymbol{\lambda}_0. \tag{5.21}$$

The successive search directions of the CG algorithm weigh the present gradient with the previous search directions using a conjugate weight. The conjugate weight is computed from the ratio of inner products as shown in (2.40), but we must here use the inner product expression for the manifold given in (2.22), as

$$\gamma_p = \frac{\langle \mathbf{G}_p - \tau \mathbf{G}_{p-1}, \mathbf{G}_p \rangle_{\boldsymbol{\lambda}_p}}{\langle \mathbf{G}_{p-1}, \mathbf{G}_{p-1} \rangle_{\boldsymbol{\lambda}_p}}. \tag{5.22}$$

Last, we compute the new search direction using the gradient and the previous search direction. However, since the old search direction may be pointing off the manifold at the new location of $\boldsymbol{\lambda}$, we must use a parallel transport of the direction. The parallel transport of the previous search direction is

$$\tau \mathbf{H}_{p-1} = \mathbf{H}_{p-1} \cos(\|\mathbf{H}_{p-1}\| u) - \boldsymbol{\lambda}_{p-1} \|\mathbf{H}_{p-1}\| \sin(\|\mathbf{H}_{p-1}\| u). \tag{5.23}$$

The new search direction is calculated using (2.41).

The complete conjugate gradient algorithm for the minimization of $J(\boldsymbol{\lambda})$ with respect to the steering vector is then

Geodesic search
$$\begin{cases} u_{\min} &= \arg\min_u J(\boldsymbol{\lambda}(u)) \text{ where} \\ \boldsymbol{\lambda}(u) &= \boldsymbol{\lambda}_{p-1}\cos(\|\mathbf{H}_{p-1}\|u) + \frac{\mathbf{H}_{p-1}}{\|\mathbf{H}_{p-1}\|}\cos(\|\mathbf{H}_{p-1}\|u) \end{cases} \tag{5.24}$$

Vector update
$$\boldsymbol{\lambda}_p = \boldsymbol{\lambda}(u_{\min}) \tag{5.25}$$

Gradient
$$\mathbf{G}_p = \nabla J_p = (-4\mathbf{F}(\boldsymbol{\theta})^T\mathbf{g} + 4\mathbf{F}(\boldsymbol{\theta})^T\mathbf{F}(\boldsymbol{\theta})(\boldsymbol{\lambda}_p \circ \boldsymbol{\lambda}_p)) \circ \boldsymbol{\lambda}_p \tag{5.26}$$

Conjugate weight
$$\begin{cases} \tau\mathbf{G}_{p-1} &= \mathbf{G}_{p-1} \text{ or } 0 \\ \gamma_p &= \dfrac{\langle \mathbf{G}_p - \tau\mathbf{G}_{p-1}, \mathbf{G}_p\rangle_{\boldsymbol{\lambda}_p}}{\langle \mathbf{G}_{p-1}, \mathbf{G}_{p-1}\rangle_{\boldsymbol{\lambda}_p}} \end{cases} \tag{5.27}$$

New search direction
$$\begin{cases} \tau\mathbf{H}_{p-1} &= \mathbf{H}_{p-1}\cos(\|\mathbf{H}_{p-1}\|u_{\min}) - \boldsymbol{\lambda}_{p-1}\|\mathbf{H}_{p-1}\|\sin(\|\mathbf{H}_{p-1}\|u_{\min}) \\ \mathbf{H}_p &= \mathbf{G}_p + \gamma_p\tau\mathbf{H}_{p-1} \end{cases} \tag{5.28}$$

5.3 Steering Vector as Posterior Probability Measures

The property that the steering vector sums to unity for the simplex method and the resultant weights are larger for the most probable targets leads us to question whether the resultant vector can be interpreted as a probability mass function for the templates. The values of $\hat{\boldsymbol{\lambda}}$ are not the posterior probabilities, but it is the case that the steering vector can be interpreted as a measure of the posterior probabilities and we can therefore infer a confidence in the final template from these values. In this section, we seek to derive expressions for the posterior probability mass function in terms of the final estimated values of the steering vector. We will demonstrate that these relations hold some value as a confidence measure on the classification. We will calculate the posterior probability from the prior probability mass function and the data probability density function assuming Gaussian noise.

5.3.1 Scalar Two Template Library

Deriving expressions for the posterior probabilities in terms of the steering vector for the full problem is intractable. We will therefore consider a simplified version where the salient issues are more easily and clearly recognized and addressed. In this section, we will derive an expression for the posterior probabilities of the templates as a function of the estimated steering vector. We will take the template library to be two scalar values on the real line. The true target is one of these scalar values and the data is the true target value with additive Gaussian noise. We can then express the data as

$$g = f + n \tag{5.29}$$

where f is the true template value and n is additive noise with a zero-mean Gaussian distribution with variance σ_n^2. The true template f is one of either $f^{(1)}$ or $f^{(2)}$ which are real scalars with $f^{(1)} < f^{(2)}$. The problem can be visualized as shown in Figure 5.5, where we see the two possible template values $f^{(1)}$ and $f^{(2)}$ and the data g lying somewhere between them. We limit the analysis to the internal region because points outside the region are not specially interesting. Specifically, the values of $\boldsymbol{\lambda}$ are constrained to lie within a convex simplex, therefore values of λ_i greater than one cannot exist. Instead, the value would settle at the vertex of the simplex. While the posterior probability can continue to increase for this data the limits on λ_i would not allow this to be reflected in the steering vector.

We define the steering vector $\boldsymbol{\lambda}$ as a vector which combines the two templates in our library into a steerable template, as

$$f_{\boldsymbol{\lambda}} = \lambda_1^2 f^{(1)} + \lambda_2^2 f^{(2)}, \text{ where } \lambda_1^2 + \lambda_2^2 = 1. \tag{5.30}$$

Using the algorithm presented previously, we are solving for the maximum likelihood

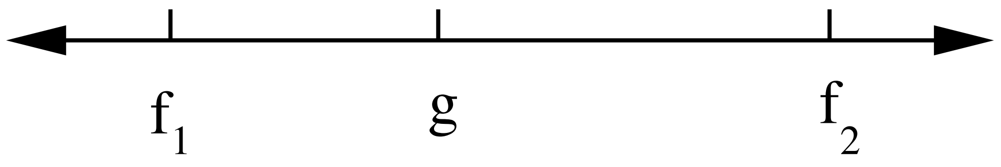


Figure 5.5: Two templates $f^{(1)}$ and $f^{(2)}$ in a one dimensional space would be represented as two points on the real line. The data g would also lie on the real line.

estimate $\hat{\boldsymbol{\lambda}}$, which is equivalent to the estimate

$$\hat{\boldsymbol{\lambda}} = \min_{\boldsymbol{\lambda}} |f_{\boldsymbol{\lambda}} - g|^2 \text{ where } \lambda_1^2 + \lambda_2^2 = 1 \text{ and } \lambda_i > 0. \tag{5.31}$$

Or we can expand the value of $\hat{\lambda}_2$ as $\hat{\lambda}_2^2 = 1 - \hat{\lambda}_1^2$, and the ML estimate becomes

$$\begin{aligned} \hat{\lambda}_1^2 &= \min_{\lambda_1} |\lambda_1^2 f^{(1)} + (1-\lambda_1^2) f^{(2)} - g|^2 \qquad 0 < \lambda_1 < 1 & (5.32) \\ \hat{\lambda}_2^2 &= 1 - \hat{\lambda}_1^2. & (5.33) \end{aligned}$$

The minimizations given result in the maximum likelihood estimators for λ_i as

$$\hat{\lambda}_1^2 = \frac{f^{(2)} - g}{f^{(2)} - f^{(1)}} \tag{5.34}$$

$$\hat{\lambda}_2^2 = \frac{g - f^{(1)}}{f^{(2)} - f^{(1)}}. \tag{5.35}$$

The posterior probability masses are the probabilities that the given template is correct given the current realization of the data. The posterior probabilities can be arrived at from the prior probability mass function and the data probability using Bayes' rule,

$$p(f^{(i)}|g) = \frac{p(g|f^{(i)})p(f^{(i)})}{p(g)}. \tag{5.36}$$

In the case of an equi-probable prior probability mass function on the templates, the prior probability $p(f^{(i)}) = \frac{1}{N_f}$ or in the present case $\frac{1}{2}$. The conditional probability

$p(g|f^{(i)})$ is the simply the probability that the noise is of the appropriate value to generate the realized data given that the true template is $f^{(i)}$. In the problem presented here, this is a Gaussian with mean $f^{(i)}$ and variance σ_n^2, or

$$p(g|f^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{1}{2\sigma^2}(g - f^{(i)})^2\right). \tag{5.37}$$

The data probability $p(g)$ is given as the prior probability weighted sums of the conditional probabilities above and is

$$p(g) = p(g|f^{(1)})p(f^{(1)}) + p(g|f^{(2)})p(f^{(2)}) \tag{5.38}$$

$$p(g) = \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(g - f^{(1)})^2\right) + \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(g - f^{(2)})^2\right). \tag{5.39}$$

Substitution the prior probabilities given in (5.37) and (5.39) into (5.36) yields the posterior probability mass for each template as

$$p(f^{(i)}|g) = \frac{\frac{1}{2}\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(g - f^{(i)})^2\right)}{\frac{1}{2}\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(g - f^{(1)})^2\right) + \frac{1}{2}\frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(g - f^{()]})^2\right)} \tag{5.40}$$

We can simplify this expression by dividing by the numerator and combining the arguments of the exponentials. We then arrive at posterior probability masses of

$$p(f^{(1)}|g) = \frac{1}{1 + \exp\left(-\frac{1}{2\sigma_n^2}((g - f^{(2)})^2 - (g - f^{(1)})^2)\right)}. \tag{5.41}$$

$$p(f^{(2)}|g) = \frac{1}{1 + \exp\left(-\frac{1}{2\sigma_n^2}((g - f^{(1)})^2 - (g - f^{(2)})^2)\right)}. \tag{5.42}$$

Substituting (5.34) and (5.35) into (5.41) and (5.42) for the noise terms $(g - f^{(i)})$

in the exponentials yields

$$p(f^{(1)}|g) = \frac{1}{1+\exp\left(-\frac{(f^{(2)}-f^{(1)})^2}{2\sigma^2}(\hat{\lambda}_1^4-\hat{\lambda}_2^4)\right)} \tag{5.43}$$

$$p(f^{(2)}|g) = \frac{1}{1+\exp\left(-\frac{(f^{(2)}-f^{(1)})^2}{2\sigma^2}(\hat{\lambda}_2^4-\hat{\lambda}_1^4)\right)}. \tag{5.44}$$

At this point we recognize that

$$\hat{\lambda}_1^4-\hat{\lambda}_2^4 = (\hat{\lambda}_1^2-\hat{\lambda}_2^2)(\hat{\lambda}_1^2+\hat{\lambda}_2^2) = \hat{\lambda}_1^2-\hat{\lambda}_2^2, \tag{5.45}$$

since the second term of the product is unity by definition. Now, we have as the expressions for the posterior probabilities

$$p(f^{(1)}|g) = \frac{1}{1+\exp\left(-\frac{(f^{(2)}-f^{(1)})^2}{2\sigma^2}(\hat{\lambda}_1^2-\hat{\lambda}_2^2)\right)} \tag{5.46}$$

$$p(f^{(2)}|g) = \frac{1}{1+\exp\left(-\frac{(f^{(2)}-f^{(1)})^2}{2\sigma^2}(\hat{\lambda}_2^2-\hat{\lambda}_1^2)\right)}. \tag{5.47}$$

It is now recognizable that the difference between the values of $\hat{\lambda}_i$ drives the posterior probabilities. That is, the larger the difference, the higher the confidence in the chosen template. This is quantifiable by (5.46) and (5.47).

Continuing, we can rearrange (5.46) or (5.47) to solve for the $\hat{\lambda}_i$ values as

$$\exp\left(-\frac{(f^{(1)}-f^{(2)})^2}{2\sigma_n^2}(\hat{\lambda}_1^2-\hat{\lambda}_2^2)\right) = \frac{1-p(f^{(1)}|g)}{p(f^{(1)}|g)} \tag{5.48}$$

$$\exp\left(-\frac{(f^{(1)}-f^{(2)})^2}{2\sigma_n^2}(\hat{\lambda}_1^2-\hat{\lambda}_2^2)\right) = \frac{p(f^{(2)}|g)}{p(f^{(1)}|g)} \tag{5.49}$$

since $p(f^{(2)}|g) = 1-p(f^{(1)}|g)$. Further manipulation produces an expression for the

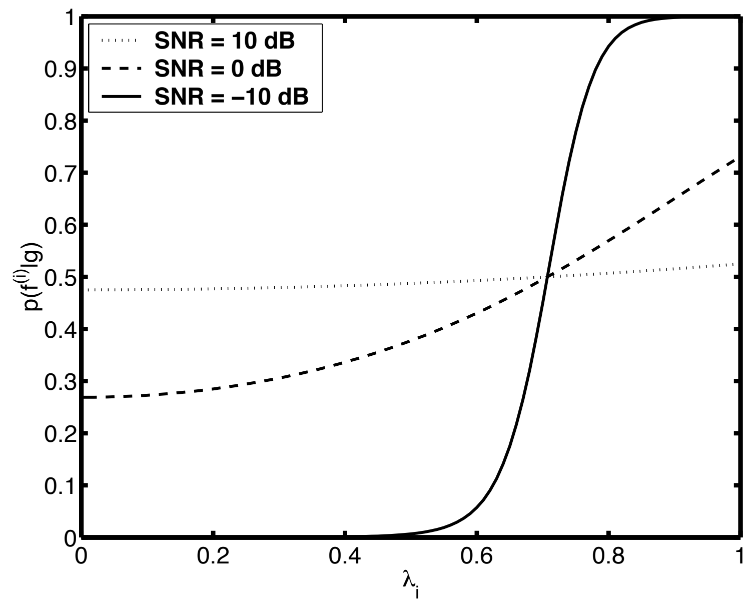


Figure 5.6: The posterior probabilities for the two-template case. The probability increases as λ_i increases.

log-likelihood ratios in terms of the difference between the values of λ_i^2 as

$$\log \frac{p(f^{(1)}|g)}{p(f^{(2)}|g)} = \frac{(f^{(1)} - f^{(2)})^2}{2\sigma_n^2}(\hat{\lambda}_1^2 - \hat{\lambda}_2^2), \tag{5.50}$$

$$= \mathrm{SNR}(\hat{\lambda}_1^2 - \hat{\lambda}_2^2). \tag{5.51}$$

The first term of the log-likelihood ratio expression is the signal to noise ratio, and the second is the difference between estimated values of λ_i. Thus the difference between $\hat{\lambda}$ estimates is directly proportional to the log-likelihood ratio. Indeed it is simply scaled by the signal to noise ratio. The log-likelihood ratio is commonly used as a decision tool in hypothesis testing [24], and thus the simple relationship between the log-likelihood ratio and the estimated $\hat{\boldsymbol{\lambda}}$ further validates the usefulness of the $\hat{\boldsymbol{\lambda}}$ vector as a decision tool for choosing templates.

In Figure 5.6, we show an example of the posterior probabilities versus the values of λ_i for the simplified two value scalar case for three values of SNR. From the plots, we see that the posterior probability of target f_i increases as λ_i increases. Further, the value passes a probability of .5 when the value of λ_i passes $\sqrt(2)$. At this point, that λ_i will be the greatest and also designates the most probable target from the template library.

5.3.2 K Template Library

The case for libraries with K templates when K is greater than 2 is more complicated and does not lead to a simple expression for the log-likelihoods, but expressions for the posterior probabilities are still possible and are still meaningful. We must still make some approximations by assuming known geometric and location parameters, but the templates are now full image templates.

Again resorting to Bayes' rule given in (5.36), we can calculate the posterior probability masses. To implement Bayes' rule, we must have expressions for the prior probabilities, the data probabilities and the conditional probabilities of the data given the true template. These are:

$$p(\mathbf{f}^{(i)}) = \frac{1}{K} \tag{5.52}$$

$$p(\mathbf{g}|\mathbf{f}^{(i)}) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{g}-\mathbf{f}^{(i)})^T(\mathbf{g}-\mathbf{f}^{(i)})\right) \tag{5.53}$$

$$p(\mathbf{g}) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \sum_{k=1} K \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{g}-\mathbf{f}^{(k)})^T(\mathbf{g}-\mathbf{f}^{(k)})\right). \tag{5.54}$$

The resultant posterior probability mass expression, simplified as before, then becomes

$$p(\mathbf{f}^{(i)}|\mathbf{g}) = \frac{1}{\sum_{k=1} K \exp\left(-\frac{1}{2\sigma_n^2}[(\mathbf{g}-\mathbf{f}^{(k)})^T(\mathbf{g}-\mathbf{f}^{(k)}) - (\mathbf{g}-\mathbf{f}^{(i)})^T(\mathbf{g}-\mathbf{f}^{(i)})]\right)}. \tag{5.55}$$

With the simplified problem above, and assuming the data is internal to the convex region with $\mathbf{f}^{(i)}$ as the vertexes, we can assume the following expression

$$g = \mathbf{F}(\hat{\boldsymbol{\lambda}} \circ \hat{\boldsymbol{\lambda}}) = \mathbf{F}\hat{\boldsymbol{\Lambda}}, \tag{5.56}$$

where for neatness of notation we have replaced the Hadamard product with the result $\hat{\mathbf{\Lambda}} = \hat{\boldsymbol{\lambda}} \circ \hat{\boldsymbol{\lambda}}$. Then the difference expressions in the argument of the exponential can be expressed as

$$\begin{aligned} \mathbf{g} - \mathbf{f}^{(i)} &= \mathbf{F}\hat{\mathbf{\Lambda}} - \mathbf{f}^{(i)} && (5.57) \\ &= \mathbf{F}\hat{\mathbf{\Lambda}} - \mathbf{f}^{(i)}\mathbf{1}^T\hat{\mathbf{\Lambda}} && (5.58) \\ &= \mathbf{X}^{(i)}\hat{\mathbf{\Lambda}}, && (5.59) \end{aligned}$$

where

$$\mathbf{X}^{(i)} = [\mathbf{f}^{(1)} - \mathbf{f}^{(i)} \cdots \mathbf{f}^{(k)} - \mathbf{f}^{(i)}]. \qquad (5.60)$$

The resulting posterior probability equation is

$$p(\mathbf{f}^{(i)}|\mathbf{g}) = \frac{1}{\sum_{k=1}^{K} \exp\left(\hat{\mathbf{\Lambda}}^T \frac{\mathbf{X}^{(k)T}\mathbf{X}^{(k)} - \mathbf{X}^{(i)T}\mathbf{X}^{(i)}}{2\sigma_n^2}\hat{\mathbf{\Lambda}}\right)} \qquad (5.61)$$

where the center term of the quadratic in the exponential is reminiscent of the SNR in the simpler case given above. It is indeed the relative SNR between two templates.

Special Case for Orthogonal Templates

While (5.61) is the most useful expression for the general posterior probabilities, we can achieve a further simplification of the posterior probability mass equations if the templates in the template library are mutually orthogonal and of uniform energy such that

$$\mathbf{f}^{(i)T}\mathbf{f}^{(j)} = \begin{cases} \sigma_f^2 & i = j \\ 0 & i \neq j \end{cases}. \qquad (5.62)$$

In this case, the matrix $\mathbf{X}^{(k)T}\mathbf{X}^{(k)}$ takes on a very simple structure (which is less than full rank). The matrix has a constant value of σ_f^2 at all locations except along the diagonal which is of value $2\sigma_f^2$. Finally, all values in the k^{th} row and k^{th} column are made zero. An example matrix with $K = 4$ and $k = 1$ would be

$$\mathbf{X}^{(1)T}\mathbf{X}^{(1)} = \sigma_f^2 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 1 \\ 0 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}, \tag{5.63}$$

which has a zero in the row and column of the subtracted template $k = 1$.

Thus when we expand out the quadratic structure given in (5.61), the relation results in the simplified posterior probability equation

$$p(\mathbf{f}^{(i)}|\mathbf{g}) = \frac{1}{\sum_{k=1}^{K} \exp\left(\frac{\sigma_{\mathbf{f}}^2}{\sigma_n^2}(\hat{\lambda}_k^2 - \hat{\lambda}_1^2)\right)}. \tag{5.64}$$

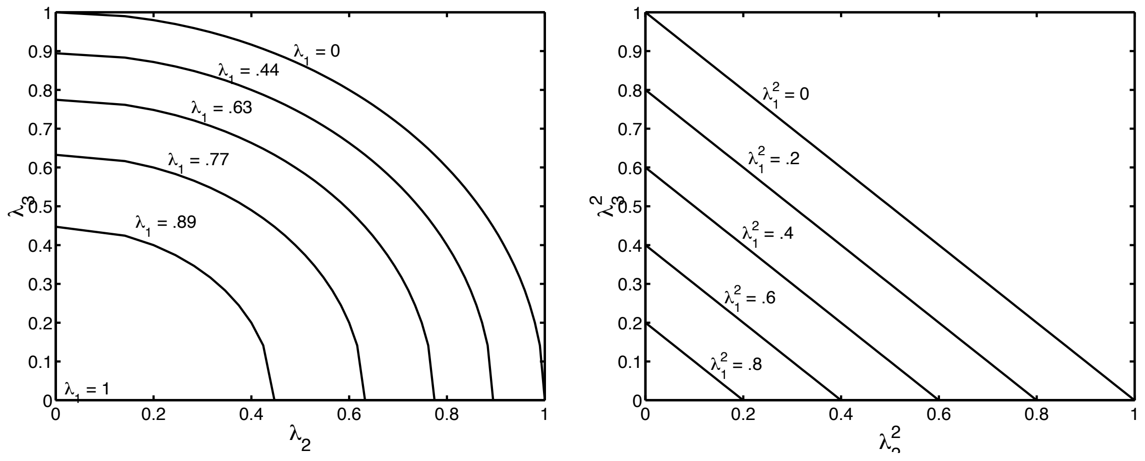
Posterior Probability Surfaces for K Template Library

The posterior probability surfaces for the K-template library are similar to those in the two dimensional case, in that we can infer a posterior probability and therefore a confidence from the values of $\hat{\boldsymbol{\lambda}}$ and the SNR. In the K-template case, however, the visualization is complicated by the higher dimensionality. We will attempt an understanding here by examining the simplest higher dimensional case of $K = 3$.

Perhaps the simplest method to visualize the posterior probability surfaces is to reduce them to the two dimensional case by "fixing" one of the coefficients of $\hat{\boldsymbol{\lambda}}$ and varying the other two within the allowed domain. If we set the value of two of the coefficients of $\hat{\boldsymbol{\lambda}}$ then the third value is of course set by the relation

$$\hat{\lambda}_1^2 = 1 - \hat{\lambda}_2^2 + \hat{\lambda}_3^2. \tag{5.65}$$

This relation is shown in Figure 5.7 in two forms. In (a) it is shown directly for the values of $\hat{\lambda}_i$ and in (b) it is shown for the squared values of $\hat{\lambda}_i$. We will usually use the mapping in (b) with squared values for $\hat{\lambda}_i$ since this leads to a linear geometry in



(a) (b)

Figure 5.7: The relationship of the three $\hat{\lambda}_i$ coefficients for the actual value in (a) and the squared value in (b).

the λ_i space.

Using the relation outlined in the previous section, we can now construct the posterior probability mass expressions for a simple three template library. In Figure 5.8, the posterior probability values are plotted for a low SNR case and a high SNR case. The charts are read by finding $\hat{\lambda}_2$ along the x-axis and finding the appropriate curve for $\hat{\lambda}_3$ and reading the corresponding posterior probability that the template $f^{(1)}$ is the correct template. Similar to what was seen in the previous section, the posterior probabilities show a more robust behavior for the estimations with respect to deviations in the coefficients of $\hat{\boldsymbol{\lambda}}$ for the high SNR case. This is shown by the width of the area with a high confidence and the sharp drop-off of the probability as another value of λ_i becomes dominant. In the low SNR case, the confidence degrades immediately as the other $\hat{\lambda}_i$ coefficients rise. Thus confidence is immediately lower for these cases.

We can expand our understanding of the surfaces by examining a surface whose axes represent the squared values of the $\hat{\boldsymbol{\lambda}}$ coefficients of the two incorrect templates. Using the relation in (5.65), it is seen that constant values of the third coefficient lie along diagonal lines in the space defined by the squared coefficients as seen in Figure 5.3. The examination here will concentrate on the region over which we can

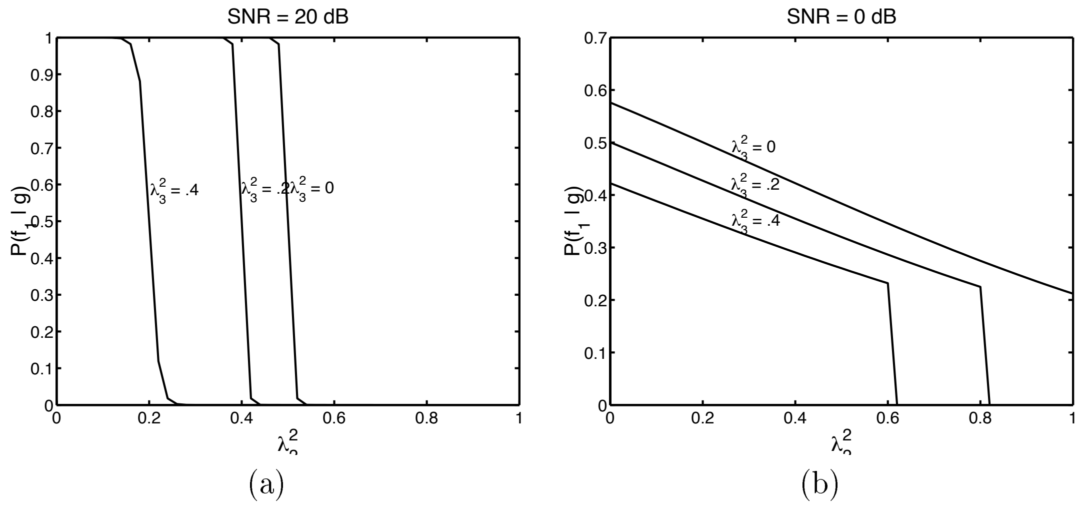


Figure 5.8: The posterior probabilities for two cases with the three template library. In (a) the SNR is high and the confidence in the correct template is also high when one value of $\hat{\lambda}_i$ is higher than the others. In (b), the posterior probabilities fall quickly as the coefficients depart from 1.

assign a high confidence.

In Figure 5.9 we have the posterior probability surface for the high SNR case. The axes are the squared values of the two coefficients of $\hat{\boldsymbol{\lambda}}$ for the templates which are not the template of interest. Constant values of the coefficient of interest $\hat{\lambda}_1^2$ lie along diagonal lines with the highest value $\hat{\lambda}_1^2 = 1$ at the origin and the lowest $\hat{\lambda}_1^2 = 0$ along a line going through the points $(1, 0)$ and $(0, 1)$. The surface plot shows that there is a large region about the origin for which the posterior probability is close to unity. This demonstrates a large region of confidence for deviations in the values of the $\hat{\lambda}_i$ coefficients. In Figure 5.10 we see the same type of surface for the low SNR case. Here, the posterior probability value is degraded to a much larger extent and yields a much smaller region over which we would be confident in the proper choice of template. This analysis agrees with that seen in the previous section.

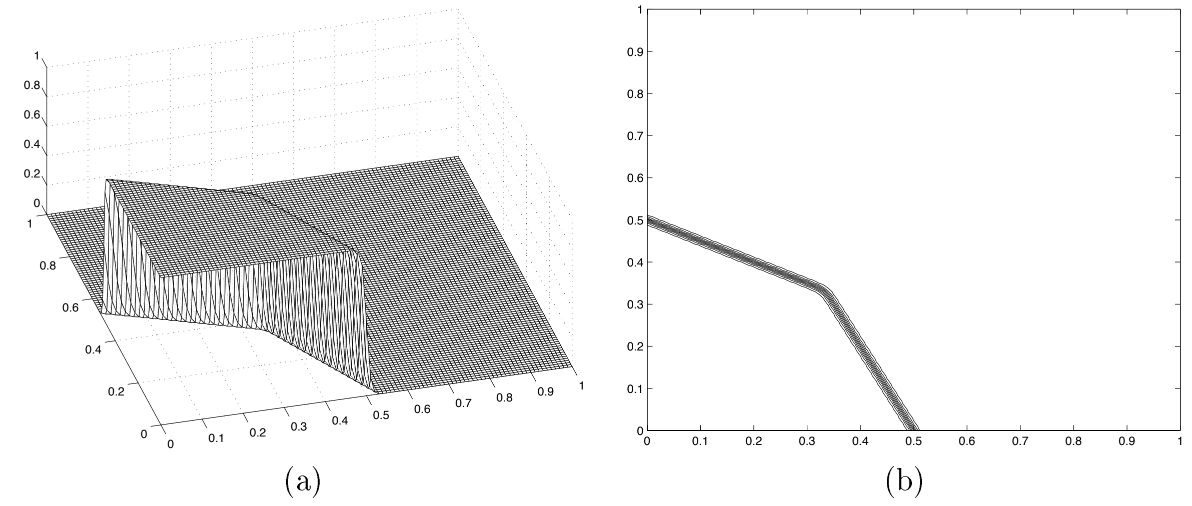


Figure 5.9: In (a) the posterior probability surface for the high SNR case and in (b) the associated contour plot. Here there is a large region of high confidence.

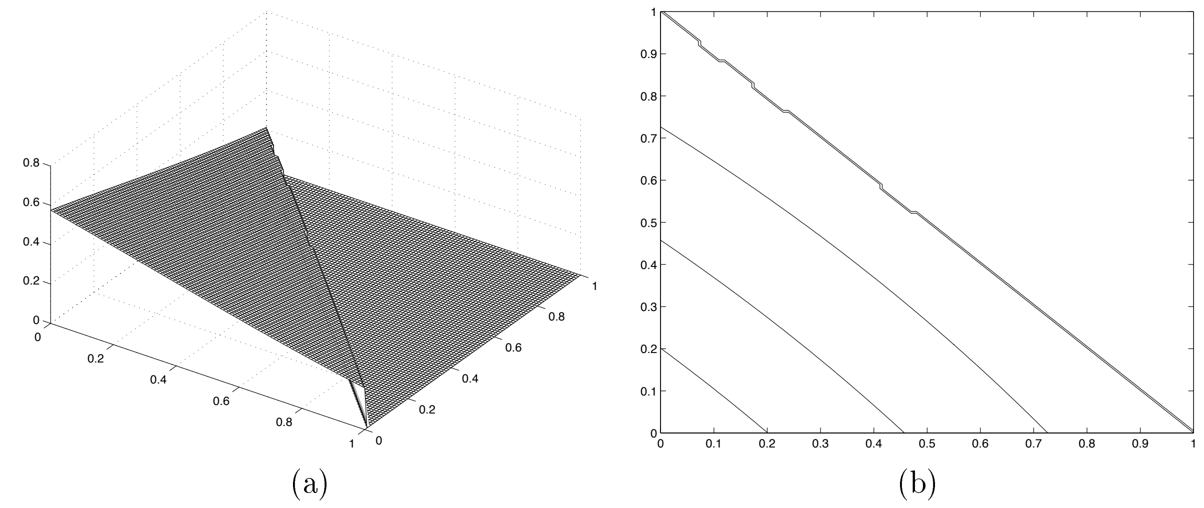


Figure 5.10: In (a) the posterior probability surface for the low SNR case and in (b) the associated contour plot. Here there is no large region of high confidence, in fact confidence degrades quickly as the coefficients depart from the origin.

5.4 Prior densities on the Steering Vector

The results of the algorithm when run with the standard cost function often leave the steering vector in an intermediate position, where the final steering vector holds values with significant amounts of several templates. In the preceding section, we examined how these values could translate into posterior probabilities of the templates. However, the discrepancy can cause errors in the estimation of the geometric parameters and can confuse the classification. Here we will examine whether it would improve classification and parameter estimation if these values were forced to finish at values which represent a single template. Hypothetically, would the parameter estimation improve if the template were forced to the *likely* true template as the algorithm enters the final stages of estimation?

For this examination we will place a prior probability on the surface of the simplex or of the hyper-sphere for where the steering vector $\hat{\boldsymbol{\lambda}}$ should settle. Ideally, the final steering vector should be one of the canonical vectors. That is, the steerable template should in the end be solely composed from a single template from the library. While placing this prior probability on the surface, however, we do not wish to prematurely force the steering vector to a solution before sufficient information has been obtained to warrant this result. Premature classification would likely result in degrading overall performance. Therefore, the prior density should, at the early stages of the algorithm, be flat and should not greatly influence the estimation of the steering vector to a significant degree. As the algorithm progresses and more information is obtained regarding the geometric parameters, the density should evolve towards a set of delta functions located at the canonical vectors.

In this section, we will adapt the previous algorithm to include a prior density into the cost function to force the final $\hat{\boldsymbol{\lambda}}$ to be a canonical vector. We will examine two possible prior densities, shown is Figure 5.11. The first density which we will examine is to include a cost based on the p-norm of the vector in the cost function. As will be shown, the p-norm for a value of $p = 1$ on the simplex and $p = 2$ on the Stiefel manifold, gives a uniform density on the cost surface, which is what is required for the early stages of the algorithm. As p is raised, the penalty on a $\hat{\boldsymbol{\lambda}}$ being a non-canonical vector increases, which forces the solution towards one of the canonical vectors. The second density which we choose to examine is to minimize the product of the distances from $\hat{\boldsymbol{\lambda}}$ to all the canonical vectors. Obviously, this penalty goes to zero when $\hat{\boldsymbol{\lambda}}$ equals a canonical vector and the distance is zero and increases for vectors away from the canonical vectors. In the case of the simplex, the Euclidean distance metric is appropriate, but in the case of the Stiefel manifold the geodesic distance metric must be used. This cost necessarily includes a weighting parameter. At the early stages of the algorithm the weighting parameter is negligible and thus the cost is flat, as the algorithm progresses, the cost becomes more significant, forcing the solution to one of the canonical vectors.

In either case, the form of the cost function will be the original cost function $J(\mathbf{r}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r}))$ plus the prior density penalty term as

$$J(\bar{\mathbf{r}}, \boldsymbol{\theta}, \boldsymbol{\lambda}; g(\mathbf{r})) \triangleq \frac{1}{2\sigma_n^2}\|g(\mathbf{r}) - f^{(t)}(\mathbf{r} - \bar{\mathbf{r}}; \boldsymbol{\theta}, \boldsymbol{\lambda})\|^2 + \Omega_p(\boldsymbol{\lambda}), \tag{5.66}$$

where $\Omega_p(\boldsymbol{\lambda})$ is the additional penalty term. The penalty term is parameterized with a weighting parameter p which in the p-norm case will be the order of the norm and in the geodesic distance case will be the weighting parameter.

5.4.1 A p-Norm Penalty

In the p-norm case, the penalty term will be the inverse of the p-norm with a varying value of p to control the amount of penalization imposed. We define the p-norm penalty as

$$\Omega_p(\boldsymbol{\lambda}) \triangleq \frac{1}{\|\boldsymbol{\lambda}\|_p} \triangleq \left(\sum_{k=1}^{K} |\lambda_k|^p\right)^{-\frac{1}{p}} \tag{5.67}$$

For $p = 1$, this cost term becomes

$$\Omega_1(\boldsymbol{\lambda}) = \left(\sum_{k=1}^{K} |\lambda_k| \right)^{-1} \tag{5.68}$$

which is of course unity for a $\boldsymbol{\lambda}$ which lies on a simplex since the sum of the λ_i values is unity, and thus this is a uniform density for the simplex algorithm. For $p = 2$, this cost term becomes

$$\Omega_2(\boldsymbol{\lambda}) = \left(\sum_{k=1}^{K} \lambda_k^2 \right)^{-\frac{1}{2}} \tag{5.69}$$

which is unity if $\boldsymbol{\lambda}$ lies on the Stiefel manifold since the sum of the squared values of $\boldsymbol{\lambda}$ is unity. Thus, at $p = 2$ this is a uniform density on the Stiefel manifold. A sequence of curves for a two-template space and varying values of p is shown in Figure 5.10(a). We see that as p increases, the density departs from unity and puts an increasingly large penalty on values of $\boldsymbol{\lambda}$ which are not associated with canonical templates, i.e. those for which a single value of λ_i is one and all others are zero.

The adaptation of the algorithm for the cost function is fairly simple. The only change is in using the new cost function for the minimization along the geodesic, and in using the gradient of the cost function for the calculation of the new gradient. Since we have specified the cost function, we need only update the gradient to have a functioning algorithm.

5.4.2 Distance Product Penalty

Similar to the case presented above, application of the algorithm to the distance product cost function is accomplished by specifying the new cost function, the derivative of the new cost function and the new gradient. The distance product penalty takes the product of the distances from the present $\boldsymbol{\lambda}$ to all the points on the surface which represent canonical templates. This penalty will therefore reduce to zero at a canonical template (where one distance will be zero) and increase as $\boldsymbol{\lambda}$ moves away from

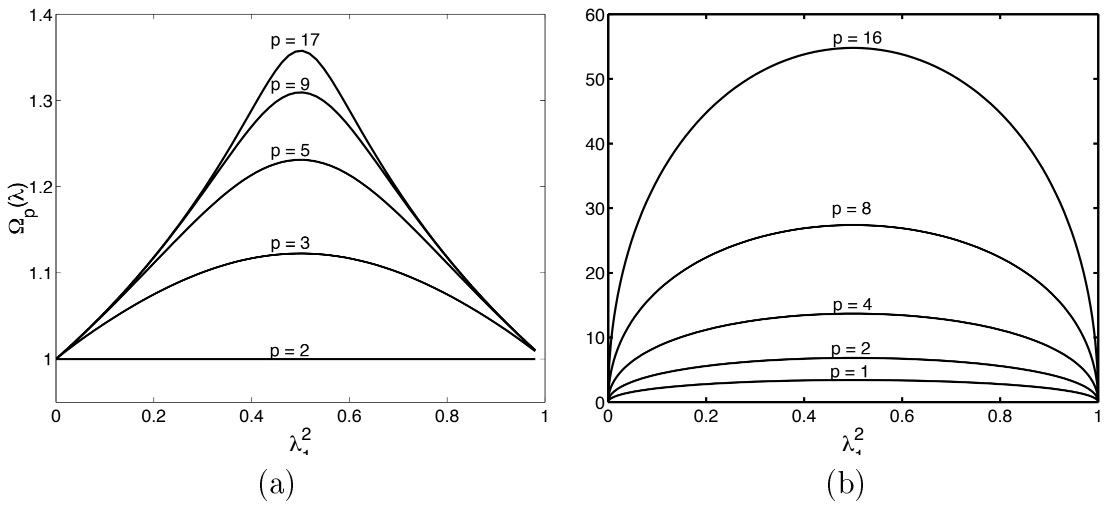


Figure 5.11: The p-norm penalty (a) and the geodesic distance penalty (b) versus one of the coefficients for the two-template case. The penalty forces the final solution towards either end of the axis.

one of these points. The amount of deviation from a uniform density is controlled through a weighting parameter p. The weighting is applied as a multiplier. We define the distance product penalty as

$$\Omega_p(\boldsymbol{\lambda}) \triangleq p \prod_{k=1}^{K} d(\boldsymbol{\lambda}, \boldsymbol{\lambda}_k) d(\boldsymbol{\lambda}, -\boldsymbol{\lambda}_k) \tag{5.70}$$

where $d(\cdot, \cdot)$ is the distance metric and $\boldsymbol{\lambda}_k$ are the steering vectors of the canonical templates. In the case of the simplex this distance metric is the Euclidean distance. In the case of the Stiefel manifold, this is the geodesic distance as shown in Chapter 2. In Figure 5.10(b) we show the distance product penalty for a series of values of p. As was seen with the p-norm penalty, the penalty increases as p increases and forces the choice of $\boldsymbol{\lambda}$ toward a canonical template.

5.4.3 Setting the p-Schedule

The final difficulty with including the prior densities is varying p in such a way as to impose the constraint at the proper time in the algorithm run. In most cases, it is only desirable to include a large cost penalty from the densities late in the estimation scheme. Thus we will begin with a small value of p and raise it slowly at the proper time. It is sensible to link the p-schedule with the t-schedule as the algorithm runs. The value of p is then computed as a scaled (and offset in the case of the p-norm penalty) value of t. We will show this in the next chapter with the examples.

5.5 Summary of Classification Algorithms

In this chapter, we have defined a general classification scheme which builds upon the parameter estimation technique which we outlined in the previous chapters. The classification algorithm simultaneously estimates the geometric parameters along with making a decision as to which template from a template library is the correct one. Further, we showed how the final values of the template steering vector can be used to compute a posterior probability for the different templates and thus a confidence bound. In the next chapter we will present results from running this algorithm.

We presented several variations on the general algorithm. We proposed that domain of the steering vector could be either a simplex in $K-1$ dimensions, or a K dimensional Stiefel manifold and discussed the advantages and implementation of each. In the case of the simplex, we expressed the necessity to use constrained optimization or bound or barrier constraints. In the case of the Stiefel manifold these constraints are not necessary. Lastly, we added the prior probability cost functions to the algorithm, the p-norm penalty and the distance product penalty.

Thus, we have now three algorithm variations with respect to the surface on which $\boldsymbol{\lambda}$ lives, they are

1. The simplex domain algorithm with boundary constraints (constrained optimization)

2. The simplex domain algorithm with barrier constraints (addition to the cost function)

3. The Stiefel domain algorithm

We can also add further variety by introducing a density onto the surface, resulting in three cost variations to the original algorithm,

1. No additional penalty
2. The p-norm penalty
3. The distance product penalty.

In the next chapter, we will explore the performance of these algorithm variations.

Chapter 6

Classification Results

In this chapter, we demonstrate some results from the classification algorithm which was presented in the previous chapter. The algorithm has been developed with three different variations depending upon the domain on which the steering vector is manipulated. Specifically, we have the simplex domain algorithm where the limits of the simplex are imposed by a hard boundary, the simplex domain algorithm with a constrictive cost barrier, and the Stiefel domain algorithm. Further, we can impose conditions upon the final solution by including a prior distribution on the cost surface which forces the final solution to be one of the templates in the template library. We examine two possible distributions, these are the p-norm and the distance product penalties.

In this chapter, we will examine the performance of these algorithms versus synthetic data. We will look specifically into the rate of correct classification, the accuracy of the geometric parameter estimates, the computational costs and the number of algorithm iterations. We will first compare the three surfaces and then examine how adding the penalty terms effects the classifications. The comparison will be done with individual runs and with Monte Carlo simulations with simple synthetic targets.

The remainder of this chapter is organized into three sections. In Section 6.1, we examine how the algorithm runs with the three different surfaces which were discussed. In this section we will discuss the dynamics of the algorithm's selection of the target and the final geometric parameters which are chosen. In Section 6.2, we will examine how the imposition of the penalty functions changes the performance. Here, we will examine the final choice of template with respect to the penalty for the algorithm running on the Stiefel manifold. In Section 6.3 we will consider more quantitative examinations of the performance. Here will examine the variance of the chosen parameters, the percentage of correct "hits" and the confusion matrices.

6.1 Comparison of the Steering Vector Domains

In this section we will present examples for the algorithm running on the three steering vector domains which were discussed in the previous chapter. The first two examples will be of the algorithm running on the simplex domain with the two different constraints. The final example will be of the algorithm running on the Stiefel domain. We will see that the final classification is similar for all examples. The advantage of the Stiefel domain is mainly in the simplicity of the imposition of the constraints.

For these examples, we examine the same problem as examined in the parameter estimation chapters. The two-peaked target is placed in a scene with 0 dB SNR. However, in the classification work, we will assume that the target is unknown. We instead only know that it is one of the templates in a template library. The algorithm has a template library which consists of three templates: the two-peaked template, a square template, and a null template. The third template represents the hypothesis that there is no target present. The three templates in the library are shown in Figure 6.1.

In Figure 6.2 we show a typical run of the algorithm for the simplex surface. The algorithm begins at the lowest value of t, which corresponds to the highest amount of smoothing or the most general template. The algorithm is initialized at an initial steering vector $\boldsymbol{\lambda}$ which is placed equidistant from the three canonical templates. Since

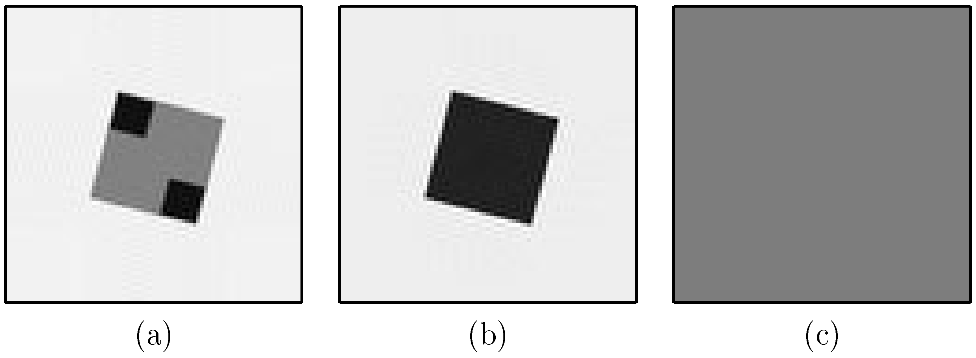


Figure 6.1: The three targets in the template library. They are (a) the two-peaked template, (b) the square template, and (c) the null template.

the initial template is a Gaussian blob and is indistinguishable across the templates in the library, the value of the steering vector is insignificant and does not effect the template. Further, the gradient with respect to the steering vector is zero so the steering vector does not change for the template $t = 0$. In the early iterations, the location and size parameters are the first to adjust to the approximate true parameter location. As the template sharpens, the template takes a slightly square shape, which drives the rotation parameter to close to the true location. As the template continues to sharpen, the $\boldsymbol{\lambda}$ parameter moves toward the correct template. However, since the line searches terminate at the edges of the simples, the steering vector frequently becomes caught on the edge. In this case, the steering vector eventually overcomes this local minima and finds the correct template. Once the correct template is chosen, the geometric parameter estimates continue to be refined. The final iterations show the template sharpen into the two-peaked template and the size, location and rotation parameters settle close to the true values. The algorithm has successfully chosen the correct template from the three and estimated its size, location and rotation correctly.

In Figure 6.3 we show the run of the algorithm for the simplex surface with the constrictive barrier for the same data. The results of the algorithm run are similar, but the pattern of estimations of $\boldsymbol{\lambda}$ is different. Instead of being caught at the edge

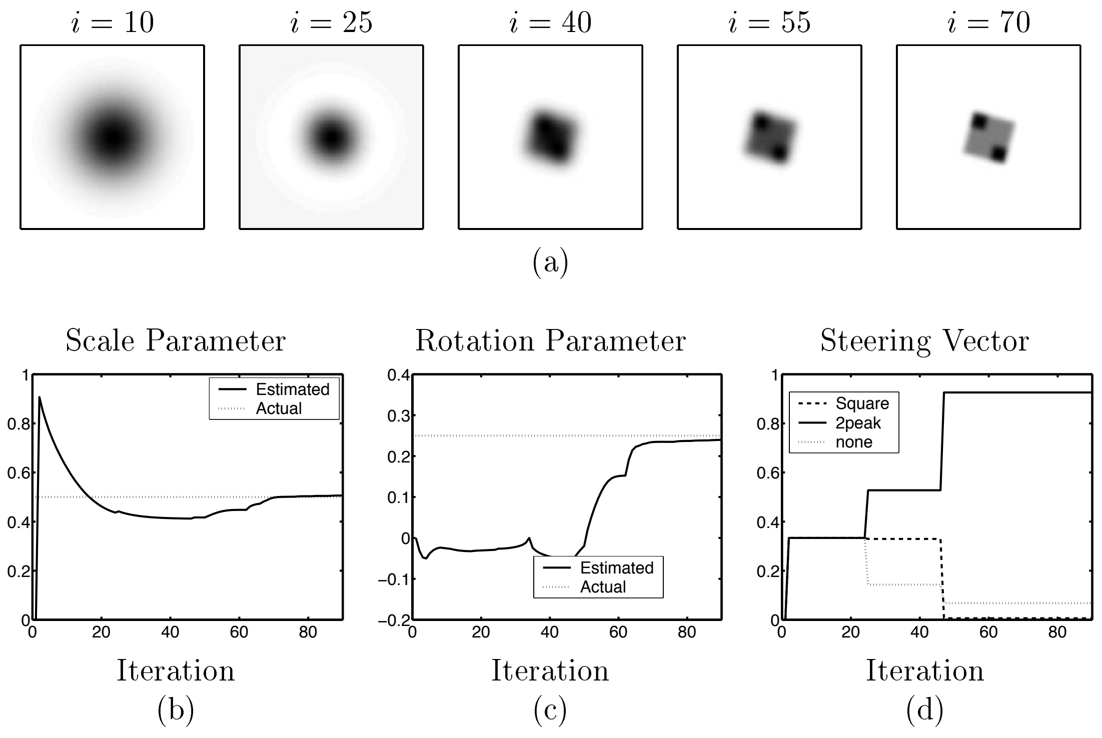


Figure 6.2: Example of the simplex domain with boundary algorithm finding position, scale, rotation and template for a two-peak target in 0 dB SNR. In (a) the estimated template is shown for 5 of the iterations. In (b) the scale parameter is shown with the true parameter. In (c) the rotation parameter is shown with the true parameter. In (d) the values of the steering vector are shown. The correct template finishes with a value of 0.9257.

of the simplex due to the hard constraint, the vector will occasionally overshoot and escape the local minimum. The final estimations of $\boldsymbol{\lambda}$ and the geometric parameters $\boldsymbol{\theta}$ are correct. This seems to be an improvement over the previous algorithm, but the creation of templates off of the simplex is bothersome. These templates do not have a logical origin. Instead of being made up of positive values (and thus positive probabilities via the equations of section 5.3), these have negative values. The last surface will address both of these conditions by avoiding the hard constraint but keeping the templates within the realm of positive probabilities.

In Figure 6.4 the Stiefel manifold domain algorithm is used for the classification. In this case, the algorithm again performs similarly and finds the true geometric parameters and the $\boldsymbol{\lambda}$ vector correctly identifies the true template. In this case, the steering vector approaches the true values in smaller steps. It becomes trapped in local minima much less frequently. This is the advantage of the Stiefel manifold over the simplex. We are able to impose the unity constraint and the positivity constraint naturally but without boundaries on the surface. The CG algorithm allows minimization on the surface without being artificially constrained with bounds or additions to the cost function. For the remainder of the examples, we will examine the performance of the Stiefel manifold since this is has the best performance and the most logical operation.

6.2 Comparison of Penalty Distributions

With the Stiefel manifold, we saw in the previous examples that the steering vector often settled on values near the correct template. However, the values did not designate exclusively one of the canonical templates. Therefore, significant amounts of another template remained in the final template. This is not always desirable, and can have an effect upon the estimates of the geometric parameters. In this section we will examine the methods for driving the steering vector to one of the canonical

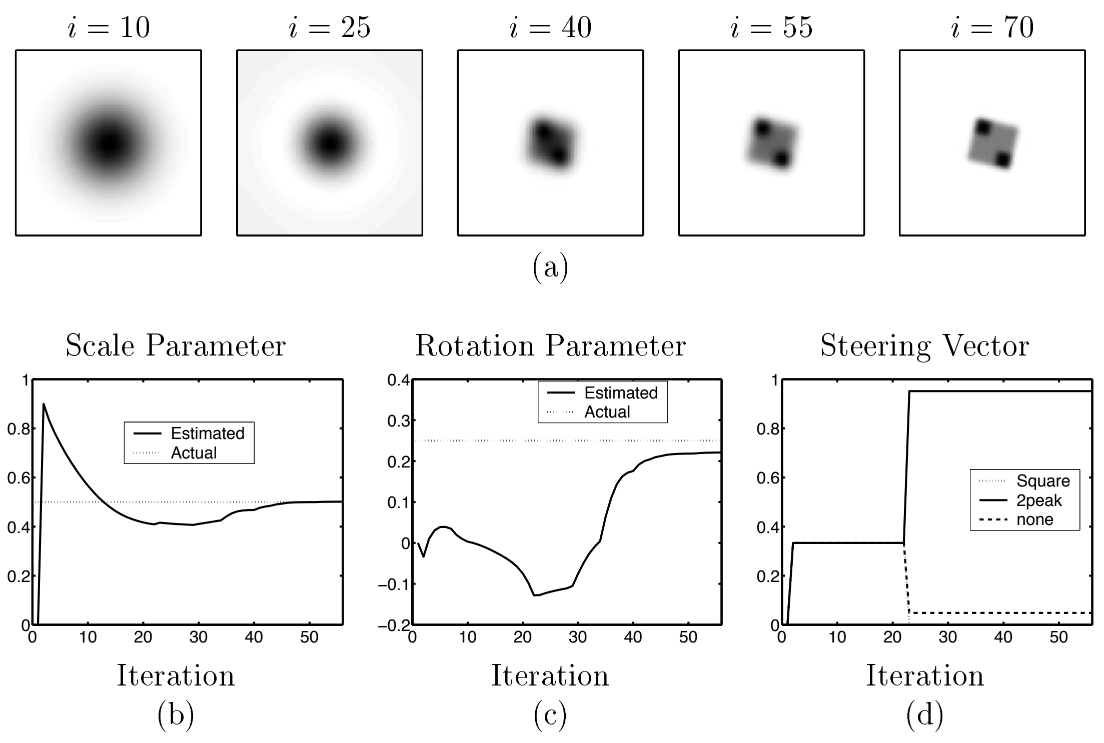


Figure 6.3: Example of the simplex with barrier algorithm finding position, scale, rotation and template for a two-peak target in 0 dB SNR. In (a), the estimated template is shown for 5 of the iterations. In (b) the scale parameter is shown with the true parameter. In (c) the rotation parameter is shown with the true parameter. In (d) the values of the steering vector are shown. The correct template finishes with a value of 0.9516.

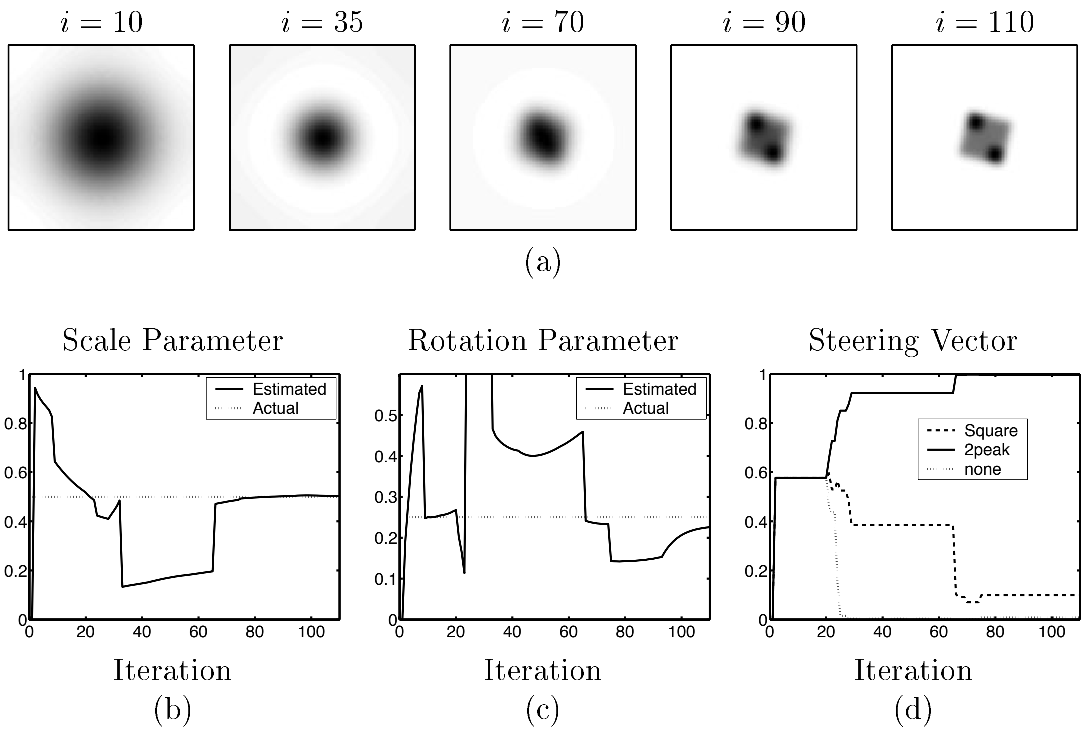


Figure 6.4: Example of the Stiefel domain algorithm finding position, scale, rotation and template for a two-peak target in 10 dB SNR. In (a) the estimated template is shown for 5 of the iterations. In (b) the scale parameter is shown with the true parameter. In (c) the rotation parameter is shown with the true parameter. In (d) the values of the steering vector are shown. The correct template finishes with a value of 0.9951.

templates by imposing a prior density upon the Stiefel manifold. The two densities investigated will be the p-norm penalty and the distance product penalty presented in the last chapter.

The first example, presented in Figure 6.5, shows the effect of using the p-norm density. We find that the p-norm density drives the steering vector towards one of the canonical vectors. This can speed convergence in the late stages of the algorithm. However, since the p-norm has a small slope in the vicinity of the canonical templates, the penalty is only effective to an extent. It does not guarantee exclusiveness to the chosen template in the last stages of the algorithm.

The second density which we examined was the distance product penalty. This penalty has a much sharper well of low penalty in the vicinity of a canonical template. This demonstrates much better performance in driving the final template to a canonical template. An example of this function is shown in Figure 6.6. We see that the final template in this case is almost exclusively composed of the two-peak template.

6.3 Monte Carlo Results

In this section we will present some more quantitative analyses of the algorithms performance using Monte Carlo simulation. We will examine the rate of correct classification, the variance of the steering vector around the correct template, the estimates of the geometric parameters and the variance of these estimates around the true parameters.

The first tool in examining the performance will be the confusion matrix for running the algorithm versus several levels of signal to noise ratio. The confusion matrix shows the percentage of runs for which the algorithm chose each resultant template versus the actual template which is in the data. For the Monte Carlo runs,

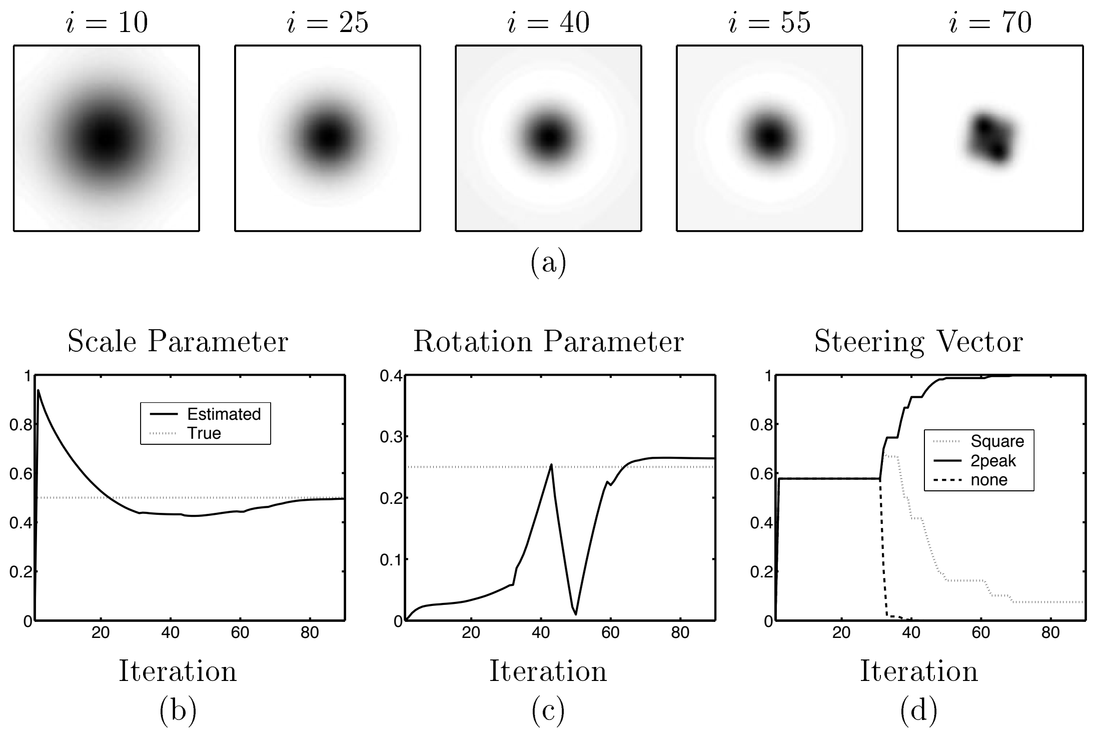


Figure 6.5: Example of the algorithm finding position, scale, rotation and template for a two-peak target in 10 dB SNR. The cost function has been augmented with the p-norm penalty term to force the steering vector to a canonical vector. In (a), the estimated template is shown for 5 of the iterations. In (b) the scale parameter is shown with the true parameter. In (c) the rotation parameter is shown with the true parameter. In (d) the values of the steering vector are shown. The correct template finishes with a value of 0.9990.

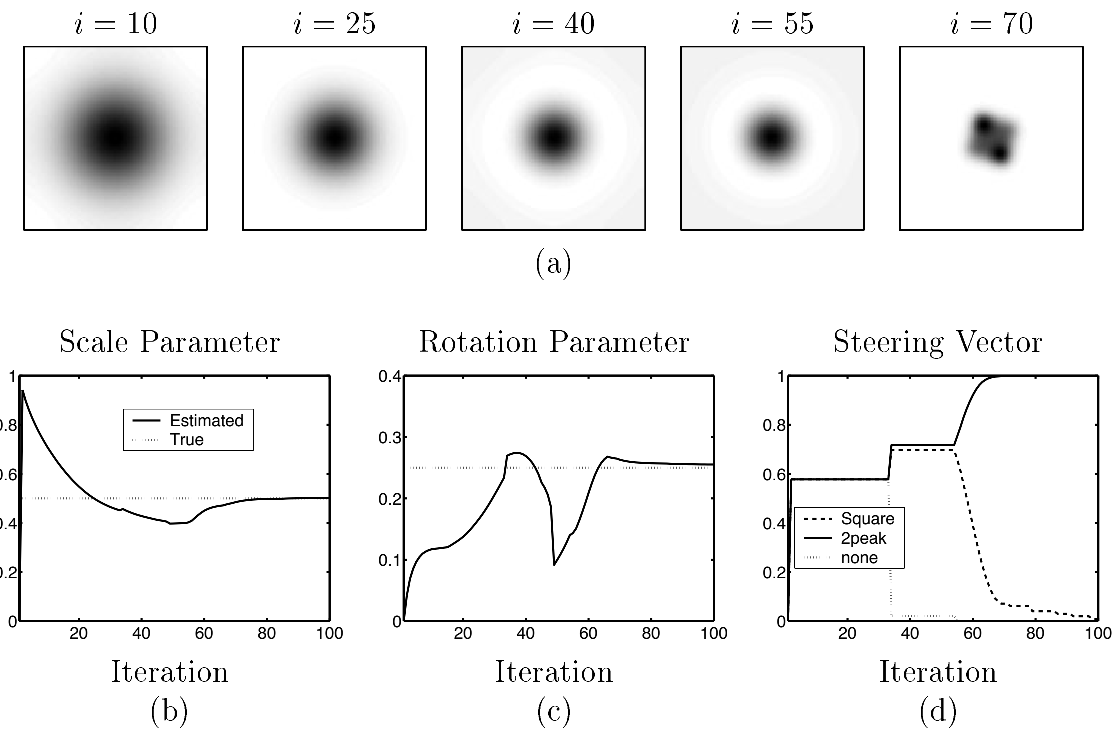


Figure 6.6: Example of the algorithm finding position, scale, rotation and template for a two-peak target in 0 dB SNR. The cost function has been augmented with the distance product penalty term to force the steering vector to a canonical vector. In (a), the estimated template is shown for 5 of the iterations. In (b) the scale parameter is shown with the true parameter. In (c) the rotation parameter is shown with the true parameter. In (d) the values of the steering vector are shown. The correct template finishes with a value of 1.000.

	SNR = 20 dB		
	2-peak	square	none
2-peak	100	0	0
square	0	100	0
none	0	0	100

	SNR=10 dB		
	2-peak	square	none
2-peak	100	0	0
square	0	100	0
none	0	0	100

	SNR=0 dB		
	2-peak	square	none
2-peak	99	1	0
square	1	98	1
none	0	0	100

Table 6.1: The confusion matrices for the Stiefel algorithm versus several levels of SNR.

the data is generated by choosing one of the templates in the library (either the two-peak, the square or no target) and adding noise at the appropriate level. The Stiefel manifold algorithm was then run to estimate the steering vector and the geometric parameters. In Table 6.1 we see the confusion matrix for three levels of SNR and 100 Monte Carlo runs for each of the templates. Values on the diagonal designate correct classification; off diagonal elements are incorrect classification. We see that the algorithm chose the correct template 100 % of the time for the high SNR case and for all three data scenarios. As the SNR decreases, several miss classifications appear. At an SNR of 0 dB we have 98 % correct classification.

Correct classification is an important statistics, but we wish to also examine how the variance of the steering vector and the geometric parameters increases for the three SNRs. Table 6.2 summarizes the means and variances of the parameters across the Monte Carlo runs. We see that the variance is low, but increasing for the three cases.

	Scale			Rotation		
Template	20	10	0	20	10	0
square	.0002	.0008	.0056	.0007	.0024	.0098
2peak	.0004	.0013	.0358	.0013	.0042	.1251

Table 6.2: The standard deviations of the geometric parameter estimates for the Stiefel algorithm Monte Carlos.

6.4 Summary

In this chapter we presented performance results from the classification algorithms. In the first section we showed the effects of the various surfaces upon the classification algorithm. The simplex algorithms were shown to work well, but with some complications. In the first, the constrained optimization, the steering vector became caught in minima against the bounds of the simplex. In the second, these minima were overcome at the cost of illogical templates. Those that included a negative amount of a template in the template library. Both these problems were overcome by implementing the Stiefel domain algorithm. The Stiefel domain algorithm avoids the troublesome hard boundaries, but does not introduce illogical templates.

With the Stiefel algorithm, we showed the various performances achieved by including a prior density on the surface. This allowed the algorithm to settle at steering vectors which were closer to the canonical template vectors. However, this destroys the confidence information which is obtained from the value of the final steering vector.

In the last section we presented the results of the Monte Carlo simulations. The MC simulations were used to construct confusion matrices for the algorithm. Here we saw good performance across a variety of SNR levels. Further, we were able to show that the algorithm retains its performance with respect to the geometric parameter estimations.

Chapter 7

Conclusions

In this thesis, we have examined the problems of target parameter estimation and target classification. We have demonstrated a unified approach to these problems which solves both through a template matching algorithm. In this Chapter, we will review the problems, solutions and contributions to parameter estimation and classification which were made in this thesis. This chapter will summarize the results and contributions of the parameter estimation work and the classification work, we will then present some ideas for future work which could be done on this topic.

7.1 Parameter Estimation

In the case of target parameter estimation, we have demonstrated a technique which uses a multiscale template library to overcome the problems associated with local minima and which is robust to noise and clutter. The method uses a template matching technique which is embedded in a multiscale iteration. As was shown in Chapter One, optimization of the likelihood surface for the template matching technique will often become trapped in local minima.

In Chapter Three, we presented a method which will overcome this problem by using a smooth approximation of the true template. The likelihood surfaces generated with the approximate templates are better behaved. However, the global minimum of this surface is much broader and therefore the estimation is less accurate. It is desirable to be able to achieve the accuracy of estimates made with the true template with the more tractable optimization available with the well behaved surface which is produced by the approximate template. In this thesis we presented an accurate and tractable optimization algorithm for performing this. The algorithm is initiated with the coarsest approximation template and thus produces the least accurate estimation. Then, by incrementally adding detail to our template and re-estimating the parameters, we achieve an accurate estimation of the parameters while remaining in a well-behaved "basis of attraction" around the global minimum and usually avoid becoming trapped in a local minima. The estimate performed with this technique retains the high accuracy available from template matching while avoiding the difficulties associated with the maximum likelihood solution.

In Chapter Four, we demonstrated the performance of this algorithm versus a simulated target and real infrared and optical images. Qualitatively, the algorithm was shown to quickly converge to the true parameters. We also presented Monte Carlo simulations of the algorithm running versus the two-peaked template. We showed that accurate estimations of the geometric parameters could be made with a low frequency of missed estimations. The missed estimations primarily occurred due to being trapped in a local minimum. We showed that by lengthening the t-schedule, the chance of being trapped in a local minimum can be reduced and increase the probability of correct parameter estimation.

7.2 Classification

In the second part of this thesis we examined how this multiscale approach could be extended to treat the target classification problem. The parameter defining target class is fundamentally different from the geometric parameters since it is a discrete index into the template library and not a continuous parameter. By introducing a steering vector, we can create a parameterized continuum of targets composed as linear combinations of the template library. This casts the classification problem as one of estimating a continuous parameter in a manner similar to the geometric parameters explored in the earlier chapters.

In Chapter Five, we demonstrated how the parameter estimation algorithm can be adapted to include the steering vector parameters. We showed how the steerable template can be smoothed in the same manner as in the parameter estimation work and thus smooth the cost surface upon which we are attempting to minimize. Then, proceeding as we did before, we can incrementally add detail to the templates, and update both the steering vector and the geometric parameters. As the estimation becomes more accurate, the steering vector will finally point to the proper template in the template library. The domain of the steering vector is most naturally represented as a simplex or the surface of a hypersphere in order to impose the unity constraint. We developed algorithms using both surfaces. Further, we showed how the steering vector can be forced to a canonical template at the conclusion of the algorithm using prior densities on the steering vector domain.

In Chapter Six, we showed the performance of these algorithms for finding the correct template from a library of templates. We demonstrated this using Monte Carlo simulations of finding a target versus a library of three templates. We presented the accuracy of the estimations and the confusion matrices which are created by the simulation.

7.3 Future Work

There are several topics which could be addressed to further the work presented in this thesis. The algorithm as presented can estimate the geometric parameters of a target which undergoes planar rotation transformations. It can also discriminate several targets from a template library under these geometric transformations. The following is a list of possible extensions to the work to further the usefulness of the algorithm:

1. The most useful extension to the algorithm would be to increase the possible geometric transformation to include three dimensional rotations. This would be an ambitious extension since it would involve incorporating a perspective mapping into the forward solution. The perspective map would profoundly change the implementation since the Fourier setting of the algorithm, which lends itself to planar rotations well, would not be easily expanded to the realm of three dimensional rotations. Further, the template models would necessarily become more complex since they would have to become three dimensional models, such as a facetted construction of the target surface.

2. As the algorithm is now constructed the background statistics are not dynamically estimated. It was shown that the algorithm can robustly estimate in the presence of background clutter, but it would be a useful extension if the clutter were also estimated. The work of Abu-Naser et. al. includes estimation of an autoregressive clutter. Other clutter models could also be formulated for appropriate settings. This would involve increasing the parameter space to include the clutter parameters. If these are continuous, they should not pose a significant impediment to algorithm construction.

3. Another useful extension would be the estimation and classification in the presence of multiple targets. This extension would necessitate either *a priori* knowledge of the number of targets or incorporate an information criteria to solve for the number of targets. Further, target location logic may become necessary to ensure that impossible solutions do not occur, such as two targets occupying the same space.

4. The algorithm calculates likelihood based on the accuracy of the match between

the data and the template. In the presence of obscuration the likelihood value would fall. However, it should be possible to estimate in the presence of obscuration by allowing lower probability scores or including some estimate of the target boundary which accounts for obscuration and appropriately credits the match.

5. In this thesis we showed the relation of the length of the t-schedule to increasing the fidelity of the parameter estimation. It would be an interesting and useful enhancement to characterize this relation. Further, it could be possible to determine an optimal t-schedule for certain situation.

6. Likewise to the t-schedule, the optimal p-scheduling could also be determined. The p-schedule should be set to impose an appropriate prior model upon the steering vector domain. This density would possibly be able to be determined by calculating the probability of the steering vector with a mismatched model.

7.4 Final Word

In this thesis we have presented a unified approach to the problems of target parameter estimation and target classification. This approach uses a multiscale algorithm to successively estimate parameters while increasing the fidelity of the model to the true template. In this way, the minimization on the likelihood surface becomes tractable while preserving the accuracy of the template match. The major contribution is a demonstration of minimizing the likelihood surface by smoothing it indirectly by smoothing the template. In doing this we have shown that the geometric parameters and the target class can be treated in common. The algorithms presented have been shown to accurately achieve estimates of target parameters and target classifications. Further work to increase the usefulness of this algorithm would be enhancement to the template models to include a perspective mapping and multiple target scenes.

Overall, we believe this is a useful approach to maximum likelihood estimation of target parameters and classification.

Bibliography

[1] A. Abu-Naser, N.P. Galatsanos, M.N. Wernick, and D. Schonfeld. Object recognition based on impulse restoration using the expectation maximization algorithm. *Journal of the Optical Society of America A: Optics and Image Science*, 15:2327, 1998.

[2] J. Ben-Arie and R. K. Rao. A novel approach to template matching by non-orthogonal image expansion. *IEEE Transactions on Circuits and Systems for Video Technology*, 3(1):71–84, 1993.

[3] J. Ben-Arie and Z. Wang. Pictorial recognition of objects employing affine invariance in the frequency domain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):604–618, June 1998.

[4] Andrew Blake and Andrew Zisserman. *Visual Reconstruction.* The MIT Press, 1987.

[5] Q. Chen, M. Defrise, and F. Deconinck. Symmetric phase-only matched filtering of fourier-mellin transforms for image registration and image recognition. *IEEE Trans. on PAMI*, 16(12), December 1994.

[6] M. Choi, N. Galatsanos, and D. Schonfeld. Image restoration based template-matching with applications to motion estimation. In *Visual Communications Image Processing '96*, pages 175–186. SPIE, 1996.

[7] M. Choi, N. Galatsanos, and D. Schonfeld. On the relation of image restoration and template-matching: application to block-matching motion estimation. In *Proceedings of the IEEE International Conference on Acoustics, SPeech, and Signal Processing 1996*, volume IV, pages 2112–2115. IEEE, 1996.

[8] L. Collatz and W. Wetterling. *Optimization Problems.* Springer-Verlag, 1975.

[9] S.A. Dudani, K.J. Breeding, and R.B. McGhee. Aircraft identification by moment invariants. *IEEE Transactions on Computers*, 26(1):39–46, 1977.

[10] A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal of Matrix Analysis and Applications*, 0, 1998.

[11] A.W.F. Edwards. *Likelihood.* The John Hopkins University Press, 1992.

[12] O. K. Ersoy and M. Zeng. Nonlinear matched filtering. *Journal of the Optical Society of America A*, 6(5):636–648, 1989.

[13] R. Fletcher. *Procatical Methods of Optimization: Volume 1, Unconstrained Optimization.* John WIley and Sons, 1980.

[14] L. R. Foulds. *Optimization Techniques: An Introduction.* Springer-Verlag, 1981.

[15] U. Grenander, M.I. Miller, and A. Srivastava. Hilbert-schmidt lower bounds for estimators on matrix lie groups for atr. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):790–802, August 1998.

[16] J.L. Horner and P.D. Gianino. Phase-only matched filtering. *Applied Optics*, 23(6):812–816, 1984.

[17] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall Inc., 1989.

[18] B. Javidi, F. Parchekani, and G. Zhang. Minimum-mean-square error filters for detecting a noisy target in background noise. *Applied Optics*, 35:6964–6975, 1996.

[19] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[20] Jan J. Koenderink. *Solid Shape*. The MIT Press, 1990.

[21] T. Kotzer, J. Rosen, and J. Shamir. Phase extraction pattern recognition. *Applied Optics*, 31(8):1126–1137, 1992.

[22] B. V. K. V. Kumar and Z. Bahri. Efficient algorithms for designing a ternary valued filter yielding maximum signal to noise ratio. *Applied Optics*, 28(10):1919–1925, 1989.

[23] B. V. K. V. Kumar and E. Pochapsky. Signal-to-noise ration considerations in modified matched spatial filters. *Journal of the Optical Society of America A*, 3(6):777–786, 1986.

[24] E. L. Lehmann. *Testing Statistical Hypotheses*. John Wiley and Sons, 1986.

[25] H. Lester and S.R. Arridge. Survey of heirarchical non-linear medical image registration. *Pattern Recognition*, 32(1):129–149, 1999.

[26] E. Marom and H. Inbar. New interpretations of wiener filtering for image recognition. *Journal of the Optical Society of America A*, 13:1325–1330, 1996.

[27] M.I. Miller, A. Srivastava, and U. Genander. Conditional-mean estimation via jump-diffusion processes in multiple target tracking/recognition. *IEEE Transactions on Signal Processing*, 43(11):2677–2690, November 1995.

[28] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[29] Mila Nikolova. Markovian reconstruction using a gnc approach. *IEEE Transactions on Image Processing*, 8(9):1204–1220, September 1999.

[30] Mila Nikolova, Jerome Idier, and Ali Mohammad-Djafari. Inversion of large-support ill-posed linear operators using a piecewise gaussian mrf. *IEEE Transactions on Image Processing*, 7(4):571–585, April 1998.

[31] L.E. Scales. *Introduction to Non-Linear Optimization.* Springer-Verlag, New York, NY, 1985.

[32] Y. Sheng and H. H. Arsenault. Experiments in pattern recognition using fourier-mellin descriptors. *Journal of the Oprical Society of America A*, 3(6):771–776, 1986.

[33] A. Srivastava. *Inferences on Transformation Groups Generating Patterns on Rigid Motions.* PhD thesis, Washington University, St. Louis, Missouri, 1996.

[34] Koichi Tanaka, Mustuo Sano, Syuichi Ohara, and Masashi Okudaira. A parametric template method and its application to robust matching. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2000.

[35] Q.M. Tieng and W.W. Boles. Recognition of 2d object contours using the wavelet transform zero-crossing representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):846–857, 1997.

[36] Zhiqian Wang and Jezekiel Ben-Arie. Model based segmentation and detection of affine transformed shapes in cluttered images. In *IEEE International Conference on Image Processing*. IEEE, October 1998.

[37] Isaac Weiss. Geometric invariants and object recognition. *International Journal of Computer Vision*, 10(3):207–231, 1993.