# GENERALIZED AREA TRACKING USING COMPLEX DISCRETE WAVELET TRANSFORM: THE COMPLEX WAVELET TRACKER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ŞENER YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2007

Approval of the thesis:

**GENERALIZED AREA TRACKING USING COMPLEX DISCRETE WAVELET TRANSFORM: THE COMPLEX WAVELET TRACKER**

submitted by **ŞENER YILMAZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department**, **The Middle East Technical University** by,

**Date:** 06 July 2007

# ABSTRACT

## GENERALIZED AREA TRACKING USING COMPLEX DISCRETE WAVELET TRANSFORM: THE COMPLEX WAVELET TRACKER

Yılmaz, Şener

Ph.D., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Mete Severcan

July 2007, 207 Pages

In this work, a new method is proposed that can be used for *area* tracking. This method is based on the Complex Discrete Wavelet Transform (CDWT) developed by Magarey and Kingsbury. The CDWT has its advantages over the traditional Discrete Wavelet Transform such as approximate shift invariance, improved directional selectivity, and robustness to noise and illumination changes.

The proposed method is a generalization of the CDWT based motion estimation method developed by Magarey and Kingsbury. The Complex Wavelet Tracker extends the original method to estimate the *true* motion of regions according to a parametric motion model. In this way, rotation, scaling, and shear type of motions can be handled in addition to pure translation.

Simulations have been performed on the proposed method including both *quantitative* and *qualitative* tests. Quantitative tests are performed on synthetically created test sequences and results have been compared to *true* data. The performance is compared with intensity-based methods. Qualitative tests are performed on real sequences and evaluations are presented empirically. The results are compared with intensity-based methods.

It is observed that the proposed method is very accurate in handling affine deformations for long term sequences and is robust to different target signatures and illumination changes. The accuracy of the proposed method is compatible with intensity-based methods. In addition to this, it can handle a wider range of cases and is robust to illuminaton changes compared to intensity-based methods.

The method can be implemented in real-time and could be a powerful replacement of current *area* trackers.

Keywords: Area Tracking, Generalized Motion Estimation, Parametric Motion Model, Optical Flow, Complex Wavelets.

# ÖZ

## KARMAŞIK AYRIK DALGACIK DÖNÜŞÜMÜ TABANLI GENELLEŞTİRİLMİŞ ALAN İZLEME: KARMAŞIK DALGACIK İZLEYİCİ

Yılmaz, Şener
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Mete Severcan

Temmuz 2007, 207 sayfa

Bu çalışmada, yeni bir *alan* izleme yöntemi önerilmektedir. Bu yöntem, Magarey ve Kingsbury tarafından geliştirilen Karmaşık Ayrık Dalgacık Dönüşümünü (KADD) temel almaktadır. KADD'nin geleneksel Ayrık Dalgacık Dönüşümüne kıyasla avantajları vardır. KADD, yaklaşık olarak yer değişimlere karşı duyarsızdır, daha iyi bir yön seçiciliği sunar ve gürültü ile aydınlık değişimlerine karşı dayanıklıdır.

Önerilen yöntem, Magarey ve Kingsbury tarafından geliştirilen KADD'ye dayalı hareket kestirim yönteminin, bölgelerin *gerçek* hareketinin parametrik bir hareket modeline göre kestirimini amaçlayan bir genelleştirmesidir. Bu sayede, düz yer değiştirmenin yanı sıra dönme, büyüme/küçülme ve kayma (*shear*) tipindeki hareketler de işlenebilmektedir.

Önerilen yöntem ile ilgili hem *nicel* hem de *nitel* testler gerçekleştirilmiştir. Nicel testler için yapay olarak oluşturulmuş test dizileri kullanılmış ve sonuçlar *gerçek* değerlerle karşılaştırılmıştır. Performans açısından görüntü-tabanlı yöntemlerle karşılaştırmalar yapılmıştır. Nitel testler gerçek diziler üzerinde gerçekleştirilmiş ve sonuçlar gözlemsel olarak yorumlanmıştır. Sonuçlar, görüntü-tabanlı yöntemlerle karşılaştırılmıştır.

Önerilen yöntemin uzun dizilerde ilgin (*affine*) hareketleri kestirmekte çok hassas olduğu ve farklı hedef görüntüleri ile parlaklık değişimlerine karşı dayanıklı olduğu gözlenmiştir. Yöntemin hassasiyeti görüntü-tabanlı yöntemlere yakındır. Buna ek olarak, daha çeşitli hedefleri güvenle işleyebilmekte ve görüntü-tabanlı yöntemlere nazaran aydınlık değişimlerine karşı dayanıklıdır.

Yöntem, gerçek zamanda çalıştırılabilmektir ve mevcut alan izleyiciler yerine güçlü bir alternatif olarak kullanılabileceği değerlendirilmektir.

Anahtar Kelimeler: Alan İzleme, Genelleştirilmiş Hareket Kestirimi, Parametrik Hareket Modeli, Optik Akış, Karmaşık Dalgacıklar.

*To my friends…*

# ACKNOWLEDGEMENTS

The path for completing a dissertation is very long and painful as well as exciting and pleasing. This work would never be finished without the help of many people.

First of all, I would like to thank my supervisor Dr. Mete Severcan for giving me the opportunity to fulfill this research and for his support, guidance, and motivation. I would also like to thank to Dr. Gözde Bozdağı Akar and Dr. Yasemin Yardımcı Çetin for their guidance and helpful discussions. I also acknowledge my thesis examining committee members for evaluating the work of this thesis. I would also like to thank Dr. Castellano in supplying me the initial material for the CDWT.

I would like thank my managers Dr. Cengiz Erbaş, Metin Sancar, and Baki Şensoy for their patience and permission. I would also like to thank all my friends in ASELSAN: Dr. Ali Kaderli, A. Erkin Arslan, Turgay Senlet, Hüseyin Özgün, Nalan Ünsoy, Burak Yenigün, Ünal Durmuş, Yüksel Serdar, Umur Akıncı, and Neval A. Yılmaz to mention only a few. Special thanks go also to all my friends in the projects I worked with for their patience and understanding.

I would like to express my gratitude to Didem Lale Merzeci, Dr. Zeynep Akalın Acar and Dr. Can Erkin Acar for their endless support, friendship, guidance and help for conducting and completing this study. Special thanks go also to Aytaç Çelik and S. Uçkun Emel for their understanding and support. Anneme, babama ve ablama her zaman yanımda oldukları için teşekkür ederim.

Many years have passed during these studies. So, please forgive me if I missed someone to mention here, as I appreciate even the slightest help and courage from anyone who love me and believe in me… And also, due to these studies, please forgive me if I could not manage to create some time. Thank you all… Herkese teşekkürler…

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AVTS | Automatic Video Tracking Systems |
| CDWT | Complex Discrete Wavelet Transform |
| CDWT-ME | Complex Discrete Wavelet Transform based Motion Estimation |
| CWT | Complex Wavelet Tracker |
| CWTF | Complex Wavelet Tracker Framework |
| DWT | Discrete Wavelet Transform |
| FLIR | Forward Looking Infrared |
| FT | Fourier Transform |
| HS | Horn and Schunck |
| IR | Infrared |
| LK | Lucas and Kanade |
| LOS | Line of Sight |
| MAP | Maximum *a posteriori* |
| ME | Motion Estimation |
| ML | Maximum likelihood |
| MS | Mean Shift Tracker |
| MSE | Mean Square Error |
| OFC | Optical Flow Constraint |
| OFE | Optical Flow Equation |
| QMF | Quadrature Mirror Filters |
| PDF | Probability Distribution Function |
| RMS | Root Mean Square |
| SSD | Sum of Squared Difference |
| STFT | Short-time Fourier Transform |
| WT | Wavelet Transform |

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathbf{x}$ | Pixel location defined as $[x_1, x_2]^T$ |
| $x_1, x_2$ | Vertical ($x_1$) and horizontal ($x_2$) components of location $\mathbf{x}$, respectively |
| $\psi(x)$ | Wavelet filter |
| $\phi(x)$ | Scaling filter |
| $m$ | Frame resolution level in the CDWT pyramid, $m_{\min} \le m \le m_{\max}$ |
| $s$ | Index of CDWT detailed subband, $\{1, 2, 3, \ldots, 6\}$ |
| $h_0, h_1$ | Wavelet basis filter pair where $h_0$ is lowpass and $h_1$ is highpass |
| $a_0, a_1$ | Amplitudes of $h_0$ and $h_1$ CDWT filter pair, respectively |
| $\omega_0, \omega_1$ | Modulation frequencies of $h_0$ and $h_1$ CDWT filter pair, respectively |
| $\sigma_0, \sigma_1$ | Window standard deviations of $h_0$ and $h_1$ CDWT filter pair, respectively |
| $L$ | Window half-length of $h_0$ and $h_1$ CDWT filter pair |
| $a_m, \hat{a}_m$ | Amplitudes of $m^{th}$ level CDWT wavelet and scaling filter pair, respectively |
| $\omega_m, \hat{\omega}_m$ | Modulation frequencies of $m^{th}$ level CDWT wavelet and scaling filter pair, respectively |
| $\sigma_m, \hat{\sigma}_m$ | Window standard deviations of $m^{th}$ level CDWT wavelet and scaling filter pair, respectively |
| $f$ | Prefilter used in the 1st level CDWT filters |
| $\Omega^{(s,m)}$ | Center (Spatial) frequency of subband $s$ of level $m$ for CDWT |
| $D^{(s,m)}$ | Detail subimage of CDWT subband $s$ of level $m$ |
| $I^{(s,m)}$ | Lowpass subimage of CDWT subband $s$ of level $m$ |

| | |
|---|---|
| $P^{(s,m)}$ | Energy of the CDWT wavelet filter of subband $s$ of level $m$ |
| $I(t)$ | Observed input image (frame) at time $t$ |
| $A(t)$ | *True* input image (frame) at time $t$ |
| $p(\cdot)$ | Probability distribution function (pdf) |
| $\mathbf{d}(\mathbf{x})$ | Displacement of pixel at location **x** |
| $W^{(m)}(\mathbf{x})$ | Set of pixels contributing to the region subpel **x** of level $m$ |
| $W^{(m)}(t_n)$ | Set of pixels within the target region on frame taken at time $t_n$ |
| $\mathbf{M}(\mathbf{x};\mathbf{p})$ | Warping function that takes the pixel at location **x** and moves it to a subpixel location $\mathbf{M}(\mathbf{x};\mathbf{p})$ with respect to frame coordinates |
| $\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p})$ | Differential warping function that takes the pixel at location **x** and moves it to a subpixel location $\mathbf{M}(\mathbf{x};\mathbf{p})$ with respect to current pixel location **x** |
| $e(\mathbf{x},t)$ | Noise at pixel **x** of frame taken at time $t$ |
| $e(\mathbf{x})$ | Differential noise at pixel **x** of frame taken at time $t$ |
| $e^{(s,m)}(\mathbf{x},t)$ | CDWT of the noise $e(\mathbf{x},t)$ in subband $s$ and level $m$ |
| $e^{(s,m)}(\mathbf{x})$ | CDWT of the differential noise $e(\mathbf{x})$ in subband $s$ and level $m$ |
| $\sigma^2$ | Noise variance in observed input frame |
| $\sigma^2_{s,m}$ | Variance of noise in subband $s$ and level $m$ |
| $\mathbf{p}$ | Track parameter vector defined as $[p_1, \ldots, p_k]^T$ |
| $\mathbf{p}^{(m)}$ | Track parameter vector for level $m$ |
| $N(t)$ | Number of pixels within the target region $W(t)$ at time $t$. |
| $k$ | Number of parameters of the track parameter vector **p** |

# CHAPTER 1

# INTRODUCTION

Tracking is an important task that humans perform quite easily in their daily lives. The eyes start tracking as soon as they are opened, and the whole process continues until they are closed again, where, of course, blinking is an exception. It is very difficult to focus the eyes without locking them onto a specific object or point. Even if there is relative motion between the eyes and the target, the eyes are able to maintain the track easily. They even keep tracking when the object undergoes major changes.

The brain creates a solution from the images acquired by our eyes. It controls the eye muscles in order to keep the target on the center of the line of sight of our eyes where the density of the receptors is greatest. It easily distinguishes the target from the background and in this way does not get affected from the background. Today's modern video tracking systems have a similar mechanism. The target motion is computed from the images obtained from an electro-optical imaging source and are converted to commands to control the azimuth and elevation angles of the platform on which the camera is mounted. In this way, the target is kept at the line of sight of the camera.

For machines, however, tracking is an enormously difficult task which is not yet solved completely despite the fact of over 30 years of effort. As the technology advances, more processing power and higher resolution sensors provides the researcher to implement more complex algorithms.

This research attempts to contribute to the area of vision-based tracking by supplying a new algorithmic base that is suitable for tracking any type of area that is selected by the operator in a direct way which is also robust and accurate. The proposed method in this thesis attempts to improve currently used area trackers especially in military applications.

This chapter follows with an overview of tracking. Next, automatic video tracking systems and their use in military applications are introduced. Then, the Complex Wavelet Tracker is presented. The Chapter ends with a summary of the contributions and an outline of the thesis.

## 1.1 Tracking

Tracking systems dates back before electro-optical imaging systems were used. The need for tracking arose in early radar systems where a beam rotates continually through 360° with a typical period of 10 seconds [13]. Two types of information are obtained from the radar: range and azimuth. If for subsequent scans more than one target is detected, then a proper association should be established among these targets in order to correctly compute their velocity vector and estimate their trajectory. In other words, a target detected in one scan should be correctly identified among other targets in subsequent scans so that its trajectory could be followed. The chances of incorrect association could be greatly reduced if the new positions of the known targets could be predicted ahead of time [13]. The Kalman filter [45] is usually used for computing the running estimates of the new states of the targets.

Unlike radars, imaging systems are passive devices which construct the image of the electromagnetic radiation entering the optics and ending in the focal plane. An example image is shown in Figure 1.1 of an infrared imaging system which is mounted on an aircraft.

Imaging systems lack the valuable range information, but provide considerably faster scanning periods and extremely higher resolutions in contrast to radar systems. A typical imaging system operates in 25 or 30 frames/second with resolutions starting from $320 \times 240$ pixels. So, the targets are not any more point sources and the image of the target consists of a set of pixels depending on the size of the target on the image plane. In this way, vision-based tracking approaches differ from the ones used for radar systems. If the target is greater than a simple blob and its shape could be identified, then, different targets could be distinguished from their corresponding 2-D projections on the frames. The relatively fast operating frequency also aids in identifying the targets that are detected in the previous frames. For most of the applications the appearance and position of the target changes only slightly. In this way, the use of predictors like the Kalman filter could be omitted as their necessity can change from application to application.

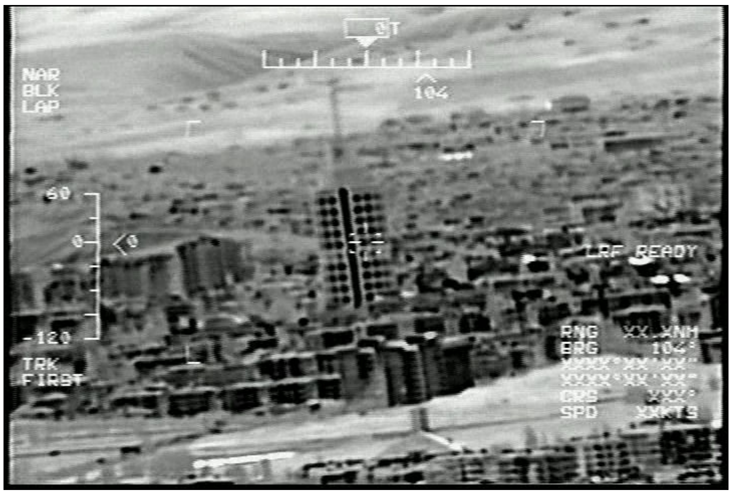


**Figure 1.1** An example image of a typical Automatic Video Tracking System (AVTS). The target is maintained in the center of the field of view of the sensor. For this example, the source is an infrared imaging source mounted on an aircraft.

The imaging system could be on a stable platform or a moving one. In addition, the sensor could be mounted on gimbals which enable the camera to turn freely on two axes. So, *ego*-motion in addition to target motion should be handled as well depending on the application. For some platforms, gimbal data, *azimuth* and *elevation* as well as platform data, *location* in space and *heading*, is available and this data could be used by the tracking algorithm. Actually, for moving platforms, if such data is not available, then, using a predictor like the Kalman filter would be difficult, as the projection of the target motion onto the 2-D image plane will be a combination of target motion and background motion (ego-motion). For such cases, if platform data is not available, ego-motion could be estimated from the video as well.

Without the range information, distinguishing the target from the background is a difficult task, since all information should be extracted solely from pixel data without range information. In this case, different segmentation methods could be applied. For surveillance applications, where the background does not change, background learning algorithms could be used to detect and extract foreground objects. However, as with such methods, tracking also requires a correspondence to be established across the frames for the specific target of interest.

Establishing a proper correspondence is another difficult problem that is dealt in the field of motion estimation. From this perspective, tracking algorithms are closely related with motion estimation methods. The problem of correspondence is an ill-posed problem [81]. To find a solution to this problem some assumptions has to be made regarding the object of interest. Motion estimation methods incorporate some smoothness constraints assuming that neighbor pixels behave similarly so that they can be handled together [42][57]. A similar assumption is also used in tracking. The object of interest is assumed to change slowly so that this change can be ignored between successive frames and that all pixels belonging to the target behave similarly [39].

However, there are other problems in tracking that should be solved together. (a) The change in the target signature is one of the problems. If a target template is used, then the template should be updated accordingly. But, this update process has its own difficulties and introduces other problems like drifting [65]. (b) Occlusions are another major problem which must be detected, so that they can be handled. There can be *partial* or *full* occlusions. Once occlusion is detected the target location could be estimated using prior information. The Kalman filter [45] or the CONDENSATION filter [44] can be used for such cases. (c) Robustness is another aspect for tracking. The method should be robust to disturbances like noise and illumination changes, it should handle a diverse set of scenarios and should maintain track for a long period of time. (d) Low or reasonable complexity is another issue. The algorithm should be able to operate in real-time with the minimum possible latency. Even a two-three frame delay in a standard video source which has a frame rate of 25 frames/sec results in more than 100 – 150 msec delay, which can be fatal for some military applications like fire control systems.

## 1.2 Automatic Video Tracking Systems

An Automatic Video Tracking System (AVTS) maintains the target at the *line of sight* (LOS) of the sensor automatically by controlling the gimbals with the aid of a tracker subsystem. AVTSs are widely used in a variety of applications including fire control, search and rescue (SAR), guidance, and surveillance. A simplified structure of a typical AVTS is shown in Figure 1.2.

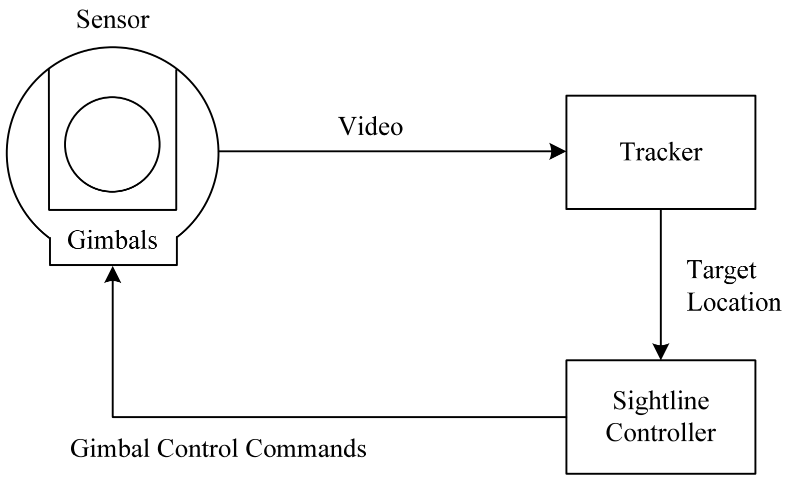


**Figure 1.2** Basic structure of a typical Automatic Video Tracking System (AVTS). The new target location is estimated from the video acquired from the sensor unit. The sightline controller adjusts the gimbals in order to keep the target within the *line of sight* (LOS) of the camera.

The purpose of an AVTS is to maintain a stable LOS from the sensor to the target automatically in the presence of target and platform motions. The target is initially located by the operator or by some other automatic target detection or recognition system (ATD/R). Once the target is *acquired*, the tracker subsystem *locks* onto it and controls the LOS automatically to maintain it on the target.

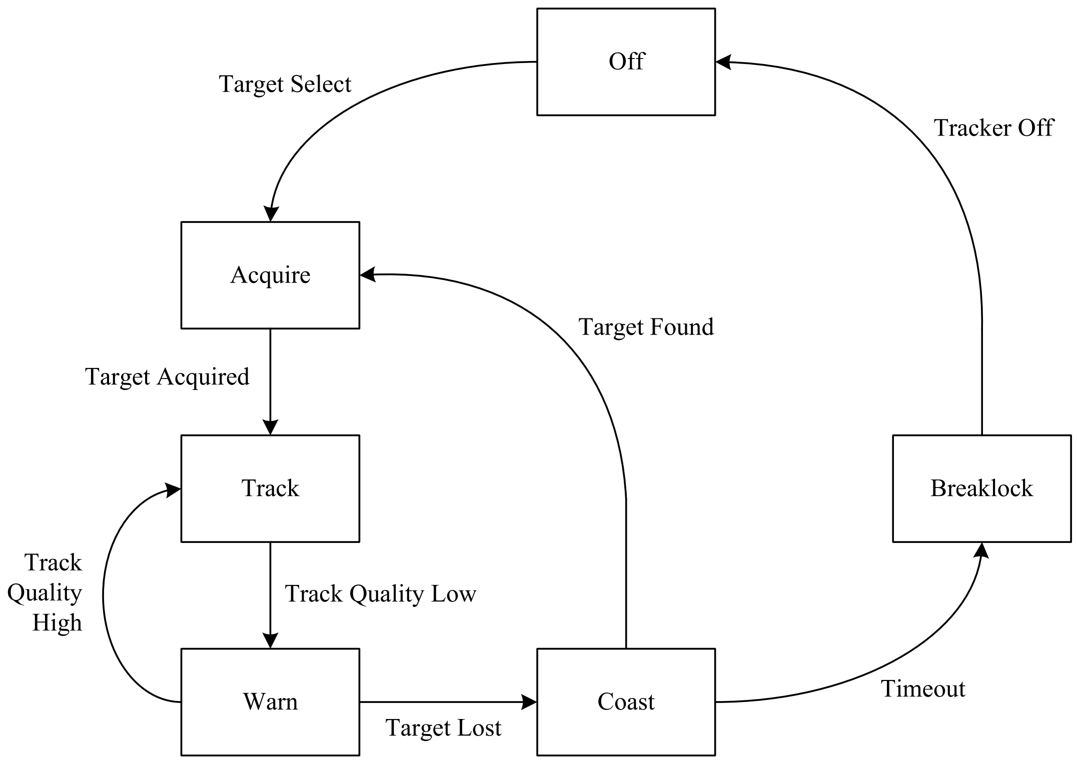


**Figure 1.3** Basic operating states of a typical Automatic Video Tracking System (AVTS).

The sensors are mounted on a gimbal control unit. The gimbals can turn the sensor unit in two axes in order to change the LOS of the sensors. The tracker subsystem computes the new target location from the video acquired from the sensor unit. The sightline controller subsystem computes the required gimbal control commands from the data obtained by the tracker subsystem.

The typical operating states for an AVTS are shown in Figure 1.3. A *track gate*, typically in a rectangular form is opened when the operator commands the tracker to select a target. After the target is selected, the tracker enters the *acquire* state where target data is collected from the video data within the track gate. After acquisition is completed the *track* state is entered. In the track state the new target location is estimated at every frame and the target data and track gate is updated accordingly. If the quality of the track gets low, the *warn* state is entered. In this state, although new location is estimated, target data might not be updated. If track quality gets high again, the *track* state is reentered, otherwise, a timeout occurs and the target is assumed to be lost. This time the *coast* state is entered where location estimation is performed using a prediction algorithm by making use of previous data in order to control the track gate, or the gimbals. If the target is found within some period of time, the tracker reenters the *acquire* state, otherwise, *breaklock* occurs and the tracker goes to the *off* state.

## 1.3 Military Trackers

In military systems, two major types of trackers are crucial: The *point* tracker and the *area* tracker. The point tracker is used to track a bounded object that can be distinguished from the background. *Centroid* tracking is used to implement the point tracker. It is based on a successful segmentation of the target from the background. It also requires that the target is completely within the field of view, so that the track gate fully encloses the entire target area. The track gate is then adapted according to the size of the target during tracking. The Centroid tracker works well for infrared video where the target is hotter than the background and can easily be segmented. For grayscale video such a distinction cannot be made easily, but for color video, if the color of the target is different from the background, the centroid tracker could be used. Otherwise, more sophisticated segmentation algorithms should be employed.

The *Area* tracker, on the other hand, could be used for many types of objects. It is usually implemented using the *correlation* tracker. It can be used for both bounded and unbounded objects, as well as to track any part of the scene. A template of the target is obtained in the acquisition phase and this template is used to search for in the subsequent frames. The search criterion is generally the correlation measure which is obtained by correlating the template with a set of search locations, thus obtaining a correlation surface. The success of the correlation tracker depends highly on the update strategy of the template. Since, the target appearance could change with time, the template should be updated accordingly, but this is a difficult task, as it is very likely that the template drifts from the actual target [65]. An improved implementation of the correlation tracker is developed by Fitts [34]. A recent review of correlation trackers can be found in [86].

For area tracking, the target may not necessarily be a distinctive object, i.e. it can be unbounded. It can be a part of an object, a part of a landscape, or a part of the scene on which the camera is aimed at. This type of tracking is essential in moving platforms like aircrafts. An example is shown in Figure 1.1 where the target is a part of a building. Due to the highly cluttered background, the target cannot be segmented easily and hence the area tracker is used.

Due to its nature, area tracking algorithms have based their approaches mostly on motion estimation methods.

## 1.4 The Complex Wavelet Tracker

This dissertation presents a new method that could be a powerful implementation of the area tracker. The proposed method is based on the motion estimation algorithm that uses the Complex Discrete Wavelet Transform (CDWT) developed by Magarey [59] in 1997. The base method is generalized to incorporate a set of pixels and allow deformations of this set according to a parametric motion model. The method operates in the CDWT domain and extracts the solution from the CDWT coefficients of the input frames.

The CDWT developed by Magarey [59] has powerful features compared to the conventional Discrete Wavelet Transform (DWT). These features are specifically intended for the use of the CDWT for dense motion estimation. The benefits of using the CDWT are (a) approximate shift-invariance; (b) improved directional selectivity; (c) sub-pixel accuracy; and (d) robustness to noise and illumination changes.

Approximate shift-invariance and improved directional selectivity is obtained by a limited redundancy of 4 to 1. The complex wavelet phase varies approximately linearly with changes in the input signal making it possible to be used for motion estimation.

A phase-based method has several advantages over intensity-based methods. Phase-based methods tend to be more robust to noise and illumination changes. Extensive tests have also shown that the CDWT based motion estimation method is superior to other methods in creating a robust, reliable, accurate and dense motion field. (See Section 6.1 for the results)

In this work we seek to improve the performance of AVTSs by extending the accuracy, robustness, and reliability of area tracking modes. For military systems, maintaining the LOS over the target is an important task, i.e. keeping the center of the target on the center of the video. From this point of view, a flow-based method has its advantages over matching-based techniques. Flow-based methods tend to be more robust to target appearance changes. We approach the problem by incorporating a parametric motion model. In this way, a more generic track gate can be used to maintain the track on the target.

This work focuses on the tracking stage only, therefore the tracker is operator initiated. The target is defined by a rectangular region called the *track gate*. This gate is then deformed according to a parametric motion model based on the projected motion of the target onto the image plane. The parameters are estimated using the CDWT coefficients of the current and previous frames.

With the Complex Wavelet Tracker we aim to address the following issues mentioned in Section 1.1. The first issue is the handling of the changes in target signature. The proposed method will take its advantage of being flow based to overcome this issue. As the parameters are estimated by making use of all target pixels, the cumulative solution will provide an adaptation to the target signature intrinsically. The second issue is robustness. Being a phase-based method, the Complex Wavelet Tracker will provide robustness, especially to illumination changes in contrast to intensity-based methods. The next issue we aim to address is low complexity. The proposed method is able to run in real-time. The issue we left out is the occlusion problem. The reason for this is that our sole aim is to develop a new algorithmic base for area tracking and thus, occlusion is left out of scope of this thesis.

The name “Complex Wavelet Tracker” is given to this algorithm since it operates purely in the complex wavelet domain. Hence, the abbreviation “CWT” will be used throughout this thesis.

## 1.5 Contributions

The major contributions of this thesis are as follows.

1. The generalization of the original Complex Discrete Wavelet Transform (CDWT) based motion estimation algorithm developed by Magarey and Kingsbury [60]. In this way, the original dense motion field estimation algorithm is applied to regions instead of individual pixels. A parametric motion model is used to describe the motion of the region. Hence, by supplying a greater support, estimation accuracy and robustness is improved, especially for low textured, smooth regions.
2. The introduction of the CDWT in the area of tracking. The development of a robust, phase-based, generalized area tracking algorithm that can be implemented in real-time. The benefits of the proposed method can be summarized as follows:
   - Robustness to noise and illumination changes. Being a phase-based method, accurate results can still be obtained even under high noise and illumination changes.
   - Sensitiveness even under low intensity patterns. It provides reliable results for low textured, smooth targets.
   - Subpixel accuracy. The solution is obtained in fractional-pel accuracy. In this way, successful tracking can be performed for long-term sequences.
   - Easy adaptation to target signature changes. Being flow-based, makes target adaptation possible in comparison to matching based methods.

- □ The inclusion of a parametric motion model allows for accurate modeling of deformations of the track gate according to the specific requirements of the application.
- □ Robustness to the type of imaging source. Grayscale, color and infrared imaging sources can be handled without the need for any changes in the algorithm and similar performance is obtained with the same parameters.
- □ Solution is obtained directly without the need for iterations.

3. The introduction of the base algorithm to other application areas such as global motion estimation, image mosaicking, image registration, and image alignment.
4. The introduction of a complete multi-target tracking framework based on the Complex Wavelet Tracker that can be further developed and extended. The framework supplies common infrastructure for the combination of different aspects like background stabilization, moving target detection, and track parameter prediction.

## 1.6 Outline of the Thesis

The thesis consists of seven chapters and one appendix. The organization of the thesis is as follows:

The next Chapter gives a review of the literature. It starts with an overview of motion estimation methods, which is followed by a review of tracking. An overview of wavelets and their use in motion estimated is provided at the end.

Chapter 3 presents the Complex Discrete Wavelet Transform (CDWT) of Magarey and Kingsbury. It starts with a description of the transform which is followed by a presentation of the properties that are important for the derivation of the proposed algorithm.

The Complex Wavelet Tracker is presented in Chapter 4. It first starts with a justification of the proposed method. Theoretical limits are defined that can be handled by the CDWT across successive frames. Second, the definition of the tracking problem is stated which is followed by a description of parametric motion models. After these preliminaries, the theoretical derivation of the proposed Complex Wavelet Tracker is presented. The derivation starts with single level parameter estimation. The solutions according to the selected parametric models are also given. This is followed by the extension of the method to a hierarchical structure.

Chapter 5 starts with some extensions to the base algorithm. Other possible application areas for the Complex Wavelet Tracker are covered next. The introduction of the Complex Wavelet Tracker Framework is given in the last section.

Chapter 6 collects the results of the simulations performed in this work. It first starts with an investigation of the original CDWT based motion estimation method. It provides a summary of the comparison results to two other motion estimation methods. Tracking simulations are presented next. First, quantitative results are given. These results are obtained from the simulations exploring the practical limits that can be achieved by the proposed method. A number of tests addressing different aspects of the algorithm are covered in the subsequent sections. Second, qualitative results are presented. These results are obtained from real tracking sequences. Discussions of the results for each type of simulations are given at the end of each section.

The last Chapter concludes with a discussion on the proposed algorithm and the results obtained from the simulations. This is followed by suggestions for further work. A conclusion is given at the end of this Chapter.

The Appendix depicts the detailed results of the comparison of the CDWT based motion estimation method to Lucas-Kanade and Horn-Schunck motion estimation algorithms.

# CHAPTER 2

# REVIEW

This Chapter reviews the literature for motion estimation, tracking, and wavelets. First, a review of motion estimation approaches are given, which is followed by a presentation of a review of tracking algorithms. A review of wavelets and their use in motion estimation is given at the end.

## 2.1 Motion Estimation

Motion estimation can be referred to as the estimation of motion in a 2-D image-plane [81] and is one of the fundamental problems in video processing. It is stated as either a correspondence or an optical flow estimation problem that is based on at least two frames, and is an *ill-posed* problem in the absence of any additional assumptions about the form of the motion [81]. It lacks from existence, uniqueness, and continuity problems. To solve the motion estimation problem, parametric and nonparametric models are used to provide additional assumptions on the structure of the motion field.

*Parametric* models aim to describe the orthographic or perspective projection of 3-D motion of a rigid surface into the image plane [81]. *Nonparametric* models impose smoothness constraints on the 2-D motion field without considering 3-D rigid motion models [81]. Nonparametric approaches can be classified as gradient-based (optical-flow-based) methods, matching-based (correspondence) methods, and frequency-based methods [52]. A detailed comparison of motion estimation methods can be found in [5].

### 2.1.1 Gradient-Based Motion Estimation

Gradient-based methods start with the assumption that the only changes in the image with time are due to the motion of 2-D intensity patterns [59]. Hence, motion estimation methods are based on the *optical flow equation* (OFE) in order to provide an estimation of the optical flow field in terms of spatio-temporal image gradients [81].

The OFE is based on the assumption that the intensity remains constant along a motion trajectory. Let $I(\mathbf{x}, t)$ denote the continuous space-time intensity distribution where $\mathbf{x}$ denotes the 2-D pixel location defined as $[x_1\ x_2]^T$. Then, this assumption can be expressed as

$$\frac{dI(\mathbf{x},t)}{dt} = 0 \tag{2.1}$$

where $\mathbf{x}$ varies with $t$ according to the motion trajectory. This equation denotes the rate of change of the intensity along the motion trajectory. To convert this expression to include the velocity vector we can use the chain rule:

$$\frac{\partial I(\mathbf{x},t)}{\partial x_1} v_1(\mathbf{x},t) + \frac{\partial I(\mathbf{x},t)}{\partial x_2} v_2(\mathbf{x},t) + \frac{\partial I(\mathbf{x},t)}{\partial t} = 0 \tag{2.2}$$

where $v_1(\mathbf{x}, t) = dx_1/dt$ and $v_2(\mathbf{x}, t) = dx_2/dt$ denote the components of the velocity vector in terms of continuous spatial coordinates. Equation (2.2) is known as the *optical flow equation* (OFE) or the *optical flow constraint* (OFC).

The OFE involves two unknowns, namely the two components of the velocity vector, $\mathbf{v} = [v_1\ v_2]^T$. In addition to this, this equation provides only a solution to the component of $\mathbf{v}$ in the direction of the gradient of $I(\mathbf{x}, t)$ at time $t$, called the *normal flow*, where the perpendicular component is unconstrained. This is known as the *aperture problem* [81].

Several suggestions have been proposed to impose a constraint on the flow field. Lucas and Kanade [57] impose a *local constraint* on the flow field by assuming that the motion vector remains unchanged over a particular block of pixels (*constant* *local flow*). Let *W* denote the set of pixels within this block, then this assumption can be expressed as

$$\mathbf{v}(\mathbf{x},t) = \mathbf{v}(t), \quad \text{for } \mathbf{x} \in W\,. \tag{2.3}$$

This model allows estimating a purely translational motion vector uniquely provided that the block of pixels contain sufficient intensity variations. Then, the problem converges to the minimization of the error in the OFE over the block of pixels *W*. Hence, the objective function can be written as

$$O(\mathbf{v}) = \sum_{\mathbf{x}\in W} \left[ \frac{\partial I(\mathbf{x},t)}{\partial x_1} v_1(t) + \frac{\partial I(\mathbf{x},t)}{\partial x_2} v_2(t) + \frac{\partial I(\mathbf{x},t)}{\partial t} \right]^2 \tag{2.4}$$

Differentiating the objective function with respect to the velocity vector, **v**, and equating to zero will yield a solution for **v** in the form:

$$\hat{\mathbf{v}} = \mathbf{Z}^{-1}\mathbf{a} \tag{2.5}$$

where

$$\begin{aligned} \mathbf{Z} &= \sum_{\mathbf{x}\in W} \begin{bmatrix} g_1^2 & g_1 g_2 \\ g_1 g_2 & g_2^2 \end{bmatrix} \\ \mathbf{a} &= \sum_{\mathbf{x}\in W} \begin{bmatrix} g_1 g_t \\ g_2 g_t \end{bmatrix} \end{aligned} \tag{2.6}$$

Here, $g_1 = \partial I(\mathbf{x},t)/\partial x_1$, $g_2 = \partial I(\mathbf{x},t)/\partial x_2$ and $g_t = \partial I(\mathbf{x},t)/\partial t$ denote the gradients. The accuracy of this method depends on the accuracy of the estimated spatial and temporal partial derivatives.

Horn and Schunck [42] imposes a *global smoothness constraint* on the optical flow field. The aim is to minimize a weighted sum of the error in the OFE and a measure of pixel-to-pixel variation of the velocity field:

$$O(\mathbf{v}(\mathbf{x},t)) = \iint_S \left[ \varepsilon_b^2(\mathbf{v}(\mathbf{x},t)) + \alpha^2 \varepsilon_c^2(\mathbf{v}(\mathbf{x},t)) \right] d\mathbf{x} \tag{2.7}$$

defined over a domain *S*, where

$$\begin{aligned}\varepsilon_b(\mathbf{v}(\mathbf{x},t)) &= \frac{\partial I(\mathbf{x},t)}{\partial x_1}v_1(t)+\frac{\partial I(\mathbf{x},t)}{\partial x_2}v_2(t)+\frac{\partial I(\mathbf{x},t)}{\partial t}\\ \varepsilon_c^2(\mathbf{v}(\mathbf{x},t)) &= \left(\frac{\partial v_1}{\partial x_1}\right)^2+\left(\frac{\partial v_1}{\partial x_2}\right)^2+\left(\frac{\partial v_2}{\partial x_1}\right)^2+\left(\frac{\partial v_2}{\partial x_2}\right)^2\end{aligned} \tag{2.8}$$

and $\alpha$ reflects the influence of the smoothness term. Iterative equations are used to minimize Equation (2.7) and obtain an estimate to the image velocity **v**.

Bruhn *et al* [14] propose a new method that combines these two approaches, namely the *combined local-global* (CGL) method. They claim that this hybrid method combines the advantages of the local Lucas-Kanade algorithm with the global Horn-Schunck algorithm.

### 2.1.2 Matching-Based Motion Estimation

Matching-based approaches are based on correspondence techniques, which are based on the identification of a set of sparse and well-identifiable features [52]. By tracking these features, an inter-frame correspondence is searched to estimate the motion of selected features on the image plane. These features can be high-level, like lines and shapes, or low-level, like corners and texture. These techniques are suitable for both motion estimation and tracking.

Block-based motion estimation is one of the most popular approaches [81]. The block-motion model can either be simple *translational* or 2-D *deformable*. The current frame is divided into blocks and the displacement of each block is estimated from the previous frame (backward estimation). For motion estimation purposes, *backward* estimation is preferred.

The *translational* block model can be characterized by a simple shift in the block of the pixels:

$$\mathbf{x}' = \mathbf{x} + \mathbf{b} \quad \text{for } \mathbf{x} \in W \tag{2.9}$$

where $\mathbf{x}'$ denotes the location of a point in the next frame and **b** denotes the shift. This is known as the *translational motion model*. Then, according to the *image-constancy assumption* [41] we can write (2.1) as

$$I(\mathbf{x},t_2)=I(\mathbf{x}+\mathbf{b},t_1)\,. \tag{2.10}$$

The block-matching method is a widely used method to estimate the displacement, where the block is searched in the previous frame by means of a matching criterion. Many different matching criteria can be used. Maximum cross-correlation, minimum mean-square error (MSE), minimum mean absolute difference (MAD), and maximum pel matching count (MPC) are among others [81]. The objective function to be minimized for the MSE criterion can be written as

$$O(\mathbf{b})=\frac{1}{N_W}\sum_{\mathbf{x}\in W}\left[I(\mathbf{x}+\mathbf{b},t_1)-I(\mathbf{x},t_2)\right]^2 \tag{2.11}$$

over the block of pixels $W$ where $N_W$ denotes the number of pixels within the set $W$. The size of the search window is selected as the maximum possible displacement that is expected between successive frames. Different search strategies can be applied to obtain the minimum. Full search, $n$-step search, or cross-search methods could be applied as well as hierarchical implementations could be used.

Besides pure translational approaches, *generalized* block-motion estimation schemes can be applied to handle other types of deformations like rotation and scaling. These methods provide superior motion tracking in the presence of rotation and zooming compared to the translational motion model [81].

The *affine* motion can be characterized by an inclusion of a deformation term, **A**, to the translational motion and can be written as

$$\mathbf{x}'=\mathbf{A}\mathbf{x}+\mathbf{b}\quad \text{for }\ \mathbf{x}\in W \tag{2.12}$$

where $\mathbf{x}'$ denotes the location of a point in the next frame. This is known as the *affine motion model*. Similarly, we can write (2.1) as

$$I(\mathbf{x},t_2)=I(\mathbf{A}\mathbf{x}+\mathbf{b},t_1)\,. \tag{2.13}$$

As in the translational motion model, the frame is divided into triangular, rectangular, hexagonal, or some arbitrary subblocks. These methods aim to track the motion of all pixels within the subblock using pixel correspondences established at the corners of the subblock [81]. Thus, it is essential that the segmentation is matching individual moving objects. Otherwise, local motion within these subblocks cannot be represented appropriately. Therefore, feature- or motion-based segmentation of the frame into adaptive meshes could be applied.

The solution of the affine motion model, however, is not as easy as in the translational motion model. Obtaining a solution via searching is no longer a trivial task. Other search strategies [81] and iterative methods [3][79] that converge to the solution could be used for this case.

### 2.1.3 Frequency-Based Motion Estimation

Frequency-based methods estimate motion by transferring the images to the frequency domain, and performing operations on the transform coefficients rather than on the image intensities.

Frequency domain techniques are mainly based on the Fourier shift theorem which states that a shift in the time domain causes a phase shift in the frequency domain. Hence, the Fourier transform of the translational motion model in (2.10) will be

$$S_{t_2}(\mathbf{f}) = S_{t_1}(\mathbf{f})\exp\{j2\pi\mathbf{f}^T\mathbf{b}\} \tag{2.14}$$

where $S_t(\mathbf{f})$ denotes the Fourier transform of image $I(\mathbf{x}, t)$. However, the Fourier transform by itself describes behavior over all time, hence, shifts that are localized to a particular time interval cannot be estimated by this method. To overcome this problem, the Short-Time Fourier Transform (STFT) is introduced. The STFT windows the Fourier basis function to provide a *time-frequency* description of the signal.

Phase correlation is derived from the Fourier shift theorem as well. The phase correlation method estimates the relative shift between the two images by means of a normalized cross-correlation function that is computed in the Fourier domain [81]:

$$C_{t_1,t_2}(\mathbf{f}) = \frac{S_{t_1}(\mathbf{f})S_{t_2}^*(\mathbf{f})}{\left|S_{t_1}(\mathbf{f})S_{t_2}^*(\mathbf{f})\right|} \tag{2.15}$$

where $C$ denotes the normalized complex-valued cross-power spectrum. Substituting (2.14) into this equation yields

$$C_{t_1,t_2}(\mathbf{f}) = \exp\{-j2\pi\mathbf{f}^T\mathbf{b}\} \tag{2.16}$$

Taking the inverse Fourier transform we obtain the phase-correlation function

$$c_{t_1,t_2}(\mathbf{x}) = \delta(\mathbf{x}-\mathbf{b}) \tag{2.17}$$

The location of the impulse yields the displacement vector **b** [81].

Estimation of motion in the Fourier domain is based on the *affine Fourier theorem* [11]. Bracewell showed that the Fourier transform of two images undergoing affine motion (2.13) will be

$$S_{t_2}(\mathbf{f}) = \frac{1}{\Delta}\exp\{j\frac{2\pi}{\Delta}[\mathbf{f_A}]^T\mathbf{b}\}S_{t_1}(\mathbf{f_A}) \tag{2.18}$$

where $\Delta = \det(\mathbf{A})$ and $\mathbf{f_A} = \Delta(\mathbf{A}^{-1}\mathbf{f})$. From the affine Fourier theorem [11], it becomes evident that the affine and translation components of the motion are separated so that the rotation, scale, and shear effect the magnitude; and translation effects the phase of the Fourier space.

Work has been conducted to obtain an estimate of the affine parameters from the frequency domain. Kumar *et al* [51] resampled Fourier magnitude spectrum and used log-polar mapping in order obtain an estimate for the affine parameters. Lucchese [58] used 1-D radial projections of the energy spectra of the segmented images in order to obtain an affine estimate. The parameters are estimated by minimizing the difference between the radial projections of the two images using the Levenberg-Marquardt minimization algorithm.

## 2.2 Tracking

Tracking is a broad research area. There are basically two types of tracking systems: *Radar tracker*s and *video tracker*s. Radar trackers operate on radar signals and try to track the trajectory of multiple objects. The target signatures are almost the same for all targets within the vicinity. The major problem is clutter. At each scan of the radar, the locations of the target points change and motion models should be incorporated in order to make correct associations between tracked targets and the detected points, especially for crossing targets. Video trackers, on the other hand, operate on video frames. Targets are represented by a set of pixels which correspond to the 2-D projection of the appearance of the target depending on the type of the imaging sensor. Visible, infrared, hyperspectral, synthetic aperture radar (SAR), and laser-radar (LADAR) are most common sensors for imaging. The tracking algorithms operate on the pixels to estimate the target state. So, the concepts of the two systems are different, and hence the approaches are different.

There are two main approaches to vision-based tracking [71][68]. The first is based on the *optical flow field*, where the flow field is analyzed to infer structure, motion, or both for the objects in the image. It is based on low-level descriptors, like intensity. The second approach is based on the *correspondences of discrete features*, where features of an object are extracted and correspondences are established in subsequent frames. This is a high-level approach which relies on image analysis and typically searches for features in an image, and then infers object motion from these correspondences.

### 2.2.1 Flow-Based Tracking

Optical flow based approaches first derive an optical flow field using the methods introduced in the previous section. This flow field is then analyzed to recover information about the dynamic scene and to make inferences about motion of 3-D objects in the scene [68].

Lucas and Kanade [57] proposed an image registration technique based on the spatial intensity gradient (See also Section 2.1.1). A type of Newton-Rhaphson iteration is used to find a match of an image block between the frames. This registration method is later used by Shi and Tomasi [79] for tracking point features. They also extended the pure translational motion model of Lucas and Kanade [57] and proposed a method for determining affine changes in a similar Newton-Raphson style minimization procedure. They proposed to use an *affine motion field* in order to have a better representation of the motion of the pixels within the feature window $W$. They defined *tracking* as the estimation of the six affine parameters given two frames [79]:

$$I(\mathbf{Ax}+\mathbf{b},t_1)=I(\mathbf{x},t_2) \quad \text{for } \mathbf{x}\in W \tag{2.19}$$

Then, the parameters **A** and **b** could be determined by minimizing the objective function

$$O(\mathbf{A},\mathbf{b}) = \iint_W [I(\mathbf{Ax}+\mathbf{b},t_1) - I(\mathbf{x},t_2)]^2 \, w(\mathbf{x}) d\mathbf{x} \tag{2.20}$$

where $w(\mathbf{x})$ is a weighting function. The objective function is minimized by differentiating it with respect to the six affine parameters and setting the result to zero. The resulting system is linearized by the truncated Taylor expansion, yielding a $6 \times 6$ linear system:

$$\mathbf{Tp} = \mathbf{a} \tag{2.21}$$

where $\mathbf{p} = [a_{11}, a_{21}, a_{12}, a_{22}, b_1, b_2]^T$ is the *affine parameter vector*. The error vector **a** depends on the difference between the two images

$$\mathbf{a} = \iint_W [I(\mathbf{x},t_1) - I(\mathbf{x},t_2)] \begin{bmatrix} x_1 g_1 \\ x_1 g_2 \\ x_2 g_1 \\ x_2 g_2 \\ g_1 \\ g_2 \end{bmatrix} w(\mathbf{x}) d\mathbf{x} \tag{2.22}$$

where $g_1 = \partial I(\mathbf{x},t)/\partial x_1$ and $g_2 = \partial I(\mathbf{x},t)/\partial x_2$ are the gradients and

$$\mathbf{T} = \iint_W \begin{bmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{V}^T & \mathbf{Z} \end{bmatrix} w(\mathbf{x}) d\mathbf{x} \tag{2.23}$$

where

$$\mathbf{U} = \begin{bmatrix} x_1^2 g_1^2 & x_1^2 g_1 g_2 & x_1 x_2 g_1^2 & x_1 x_2 g_1 g_2 \\ x_1^2 g_1 g_2 & x_1^2 g_2^2 & x_1 x_2 g_1 g_2 & x_1 x_2 g_2^2 \\ x_1 x_2 g_1^2 & x_1 x_2 g_1 g_2 & x_2^2 g_1^2 & x_2^2 g_1 g_2 \\ x_1 x_2 g_1 g_2 & x_1 x_2 g_2^2 & x_2^2 g_1 g_2 & x_2^2 g_2^2 \end{bmatrix}$$
$$\mathbf{V} = \begin{bmatrix} x_1 g_1^2 & x_1 g_1 g_2 \\ x_1 g_1 g_2 & x_1 g_2^2 \\ x_2 g_1^2 & x_2 g_1 g_2 \\ x_2 g_1 g_2 & x_2 g_2^2 \end{bmatrix} \tag{2.24}$$
$$\mathbf{Z} = \begin{bmatrix} g_1^2 & g_1 g_2 \\ g_1 g_2 & g_2^2 \end{bmatrix}$$

Note that, the pure translational motion model corresponds to a subset of these equations:

$$\mathbf{Zb} = \mathbf{e} \tag{2.25}$$

where **e** collects only the last two entries of **a** in Equation (2.22).

Hager and Belhumeur [39] proposed a region tracking method using a parametric model for geometry and illumination. They started with the same goal as Shi and Tomasi [79] algorithm, which is also a generalization of the Lucas-Kanade algorithm, defined in Equation (2.20) with a change of the affine motion model with a more generic warping function, **M(x;p)**. They defined this function as the *parametric motion model*; and *tracking* as recovering the parameters **p** throughout the input frame sequence [39].

At the beginning, the target is defined by a rectangular region at time $t_0$. Then, for an arbitrary time $t > t_0$, as the geometry of the target would change, the motion of this rectangular region is defined by this parametric motion model with parameters $\mathbf{p}(t)$ with respect to $t_0$. In this way, the problem is converted to the estimation of a vector of offsets, $\Delta\mathbf{p}$. Then, at each iteration, **p** will be updated as

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}. \tag{2.26}$$

Baker and Matthews [3] addressed this method as the *inverse additive* algorithm and proposed the *forwards compositional* and *inverse compositional* algorithms to overcome such limitations. They reformulated the iteration steps such that the algorithm solves for an incremental warp $\mathbf{M}(\mathbf{x};\Delta\mathbf{p})$ rather than an additive update to the parameters $\Delta\mathbf{p}$:

$$\mathbf{M}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{M}(\mathbf{M}(\mathbf{x};\Delta\mathbf{p});\mathbf{p}). \tag{2.27}$$

They claim that although the computational cost is equivalent to Hager-Belhumeur algorithm, the restriction on the allowed warp is loosened. Together with the LK algorithm, which they refer to as the *forwards additive* algorithm, they showed that these four methods yield the same results as a means of convergence [3]. They added that in case of additive noise, this similarity breaks in favor of forwards algorithms which have performed slightly better.

### 2.2.2 Feature-Based Tracking

Feature-based tracking is based on the correspondence of discrete features of an object in one frame with those features in a subsequent frame. Unlike flow-based methods, this approach relies on discrete features, instead of processing the whole target area. A feature can be any easily observable characteristic of the object being tracked. Any measurable relationship in an image could be used as features. For tracking, commonly; points, edges, corners, textures, and regions are used as features. After the features are extracted, inter-frame correspondences are established among them. Constraints can also be established based on the assumptions on the motion of the object to be tracked [72].

Computation of new invariants has been introduced recently for using as features in tracking [71]. Instead of standard features such as corners and lines, projective invariants as well as parametric measures that are insensitive to both changes in geometry and illumination can be used which are image features independent from camera position.

For **Point Tracking**, Davies *et al* [27] presented a Kalman tracking algorithm that can track a number of very small, low contrast objects. The objects are detected by the Daubechies wavelet filter and segmented using morphological operations. Due to the small size of the objects, they are represented by points. Multiple track hypotheses are created and a Kalman filter is used to track the targets. Caefer *et al*

[15] used a triple temporal filter (TTF) with six input parameters for detecting and tracking point targets in consecutive frame data acquired with staring infrared cameras. The generic form of the filter weights the data of each pixel in time by a zero-mean damped sinusoid.

*Corners* are used as good feature points in tracking. Nassif [67] used corner tracking algorithm as one of the two methods he used in his tracking framework. Behrad *et al* [7] used future points selected from the edge of the target. These feature points are then matched with points in the region of interest in the next frame using fuzzy-edge-based feature matching for tracking.

A **Blob Tracking** algorithm computes the state of a target based on pixelwise operators which identify every pixel in a tracking window as being either a part of or not a part of the target object [84]. Blob tracking is adequate for tracking in structured environments where one has considerable control over the visual environment and the object to be tracked. Its advantages include speed and simplicity.

For **Edge Tracking**, *lines* and *edges* are popular choices as features due to their robustness to illumination variance and ease of extraction from the image. Line extraction is less sensitive to noise than point extraction, and line correspondence is usually an easier problem to solve than point correspondence. The edge tracking procedure is divided into two stages: feature detection and state updating. In the detection stage, rotational image warping can be used to acquire a window which, if the prior estimate is correct, leads to an edge that appears vertical within the warped window [84].

For **Contour Tracking**, Schoepflin *et al* [75] proposed an algorithm that applies a sequence of hierarchical deformations to find a match for a dynamic template in a video frame. They use hierarchy of separate deformation stages: global affine deformations, local (piecewise) deformations, and snake-based continuous deformations. A *snake* is an active contour adapted by minimizing its energy. The internal energy of the snake seeks to maintain the shape of the contour, opposing deformation, and its external energy, on the other hand, seeks to place the snake along the edges of the image defined between an object and its background [36].

Koller *et al* [49] employed a *contour tracker* based on intensity and motion boundaries to track multiple 3-D objects in a known environment with occlusion reasoning. They assumed that the motion of a contour enclosing the image of a vehicle to be well describable by an affine motion model with a translation and a change in scale. The contour is based on motion and intensity boundaries. They used a closed cubic spline with twelve control points to approximate the extracted convex polygon.

In **Region-Based Tracking**, a *region*, often defined as a maximal homogeneous image patch, is tracked as a feature [68]. Vigus [85] used region split and merge method and Kalman filtering to track video objects. He assumed video objects to be groups of image pixels coherent spatially as well as in their values of luminance. Lim *et al* [54] proposed a semantic object tracking scheme based on re-estimation of a morphological region-based segmentation approach for tracking. Previous work of Lim [55] with Ra was based on a three-step boundary projection scheme: object boundary projection, uncertain area extraction, and boundary refinement. Region growing is utilized to merge uncertain pixels and the pixels near the uncertain area. Region growing is based on color information.

Bremond and Thonnat [12] tracked moving regions detected from a fixed camera using a five-point dynamic target model. Cohen and Medioni [20][21][22] used moving regions to detect and track moving objects in airborne image sequences. They detected moving regions by estimating egomotion and represented them by a graph in order to create relations among them. Hwang *et al* [43] used spatial segmentation for tracking independently moving objects, called *video object planes* (VOPs) in visual coding. He performed spatial segmentation using distributed genetic algorithms. Once moving objects are extracted from each frame, object tracking is automatically performed by tracking on the spatial segmentation level.

In **Template-Based Tracking** a stored reference template is matched to a region of the image. The reference is either a region taken from the scene itself, or a pre-supplied target template [84]. A *feature template* may also be used to detect a feature in an image. A feature template contains some representation of the feature and is compared against portions of an image to locate that feature in the image [68]. Lipton *et al* [56] used adaptive templates for tracking targets. Moving regions are detected by temporal differencing and then used as training templates for the tracker. The template is updated adaptively by merging previous instance with current information. Canals *et al* [16] used a multi-block technique for tracking deformable objects. Since a square block cannot cover the whole object without including nontarget points in the model, they chose a multi-block solution in which several square blocks are used to model the target. The size of each subblock is updated depending on its size at the previous moment and on the evolution of the distances among the subblocks.

*Deformable templates* are commonly used as the 2-D model of the object. If the object is known, one can design a deformable template to appropriately constrain the object segmentation. Deformable templates can model a wide variety of objects, ranging from those with known geometry to instances where the allowable deformations are unknown, in which case very flexible templates, such as snakes [76] must be used.

Cohen and Medioni [20][21][22] used dynamic templates for tracking objects. A dynamic template is obtained for each detected region by applying a median filter over the last five detected frames of the region. Bremond and Thonnat [12] used a five-point dynamic target model to track moving regions. The tracking is performed in a prediction-matching-update loop. The target trajectory is represented by a polygonal approximated.

In **Model-Based Tracking**, most of the approaches assume that there is a model given as known *a priori* information. The goal is to find a match that minimizes a cost function between the model and image features. A work on model-based tracking of articulated objects is conducted by Nickels [68]. The assumption is that the appearance model and the kinematics structure of the object are given. The observations from the image and the knowledge of the previous configuration of the object are combined to estimate the new configuration using the Extended Kalman filter. The tracker observers a monocular grayscale image of the scene and combines information gathered from this image with knowledge of the previous configuration of the object to estimate the configuration of the object at the time the picture was taken.

Nassif [67] chose a model-based tracking algorithm for one of the two methods he used in his framework for tracking. He *et al* [40] presented an object tracking method which uses the 2-D Gabor wavelet transform (GWT) and a 2-D golden section algorithm. The feature points are stochastically selected based on the energy of their GWT coefficients. The global placement of the feature points is determined by a 2-D mesh whose feature is the area of the triangles formed by the feature points. In order to find the corresponding object in the next frame, the 2-D golden section algorithm is employed. Nummiaro *et al* [69] proposed an adaptive color-based particle filter for tracking objects. A particle filter tracks several hypotheses simultaneously and weights them according to their similarity to the target model. The proposed method adds an adaptive appearance model based on color distributions to particle filtering. As multiple hypotheses are processed, objects are able to be tracked in case of occlusion and clutter. Foresti [33] proposed an integrated framework for recognition and tracking. He used the statistical morphological skeleton (SMS) of the object as a shape descriptor and compared it with other model objects stored in a database. He then derived observable quantities from the detected SMS and applied an extended Kalman filter to them to track the moving object.

Comaniciu [24] developed the **Mean Shift Tracker** where he used a *target model* that is obtained from image statistics. The statistics are also weighted with an isotropic kernel according to its location on frame. This kernel induces spatially smooth similarity functions suitable for gradient-based optimization. In this way, the target localization problem is formulated using the basin of attraction of the local maxima [23]. The Bhattacharyya coefficient is used as a similarity measure and the mean shift procedure is used to perform the optimization.

The mean-shift procedure [24] can be summarized as follows: First, the target model and candidate models are created using an *m*-bin histogram. Then, the ratio histogram is obtained by normalizing target bins with candidate bins. This ratio histogram is then used to create the backprojected image where the mean shift iterations are performed. Scale adaptation [23] is performed by running the algorithm three times for different target scales: smaller, equal, and larger size; and by selecting the one resulting in the largest Bhattacharyya coefficient. Filtering [23] can be used to avoid oversensitive scale adaptation.

## 2.3 Wavelets

The wavelet transform has become a popular tool for image analysis in the recent decade. Unlike the Fourier transform, it provides time-frequency localization of signals. It may be regarded as equivalent to filtering the input signal with a bank of bandpass filters, whose impulse responses are approximately scaled versions of a *mother wavelet*. This section gives a brief introduction of the wavelet transform followed by a review of the use of wavelets for motion estimation.

### 2.3.1 The Continuous Wavelet Transform

The *Continuous Wavelet Transform* of a one-dimensional signal $I(t)$ can be defined as

$$CWT\{I(t)\} = D(\tau,\alpha) = \int_{-\infty}^{\infty} I(t)\psi_{\tau\alpha}(t)dt \tag{2.28}$$

where $D$ is used for the wavelet transform referring to *detailed* subbands and $\psi_{\tau\alpha}(t)$ represents a set of wavelets, generated from the mother wavelet $\psi(t)$ by a change of scale and a translation in time. The scale of $\psi(t)$ is conventionally 1, and that of $\psi_{\tau\alpha}(t)$ is $\alpha > 0$. The function $\psi(t)$ is conventionally centered around 0, and $\psi_{\tau\alpha}(t)$ is then centered around $\tau$. Thus we have

$$\psi_{\tau\alpha}(t) = \frac{1}{\sqrt{\alpha}}\psi\left(\frac{t-\tau}{\alpha}\right) \tag{2.29}$$

where $\alpha, \tau \in \mathbf{R}$. The scale factor $\alpha$ represents the scaling of the wavelet function where a large scale mean stretched wavelets for global views and a small scale means shrunk wavelets for detailed views.

### 2.3.2 The Discrete Wavelet Transform

The Continuous Wavelet Transform is highly redundant, and hence, the scale and translation parameters could be sampled to obtain the *Discrete Wavelet Transform* (DWT). Usually, a dyadic sampling is used for $\alpha$ and a linear (proportional to the scale) sampling for $p$, i.e., $\alpha = 2^m\alpha_0$ and $\tau = k\alpha$, where $m, k \in \mathrm{Z}$. Then, the DWT can be written as a set of integer-indexed coefficients:

$$D_{mk} = \int_{-\infty}^{\infty} I(t)\psi_{mk}(t)dt \tag{2.30}$$

$$\psi_{mk}(t) = 2^{-m/2}\psi(2^{-m}t - k) \tag{2.31}$$

with $m, k \in \mathrm{Z}$. When the signal is also sampled in time ($I = I(n)$, $n \in \mathrm{Z}$), which is the case in most signal processing applications, we obtain the *Discrete-Time Wavelet Transform* (DTWT) defined as

$$D^{(m)}(n) = \sum_k I(k)\psi^{(m)}(2^m n - k) \tag{2.32}$$

A *subband decomposition tree* structure could be used to implement the DTWT. Referring to Figure 2.1, a pair of halfband filters, $\{h_0, h_1\}$ could be used for the analysis. The highpass filter $h_1$ provides the detail coefficients $D^{(\mathrm{m})}$ and the lowpass filter $h_0$ creates the coarse approximation of $I(t)$. The equivalent wavelet filters $\psi^{(m)}(n)$ of Equation (2.32) can be written in terms of $h_0$ and $h_1$ as follows:

$$\Psi^{(m)}(z) = H_1(z^{2^{m-1}})\prod_{k=0}^{m-2} H_0(z^{2^k}) \tag{2.33}$$

where the Z-transform is used to describe the digital filters. To represent the lowpass outputs $I(m)$ the *scaling filters* $\phi(n)$ are used which are defined in a similar manner as

$$\Phi^{(m)}(z) = \prod_{k=0}^{m-1} H_0(z^{2^k})\,. \tag{2.34}$$

For images, the wavelet transform should be extended to 2-D. This is accomplished by extending the subband decomposition tree to implement a *separable* wavelet transform [64]. Hence, a 2-D DWT[1] applies the 1-D building block, shown in Figure 2.1, first to the rows, then along the columns of the resulting two half-sized subimages, resulting in three detailed subimages $D^{(s,m)}$ and one lowpass subimage $I^{(m)}$. This structure is shown in Figure 2.3. Here, $s$ refers to the subband, 1 through 3, and $m$ represents the level of the decomposition.

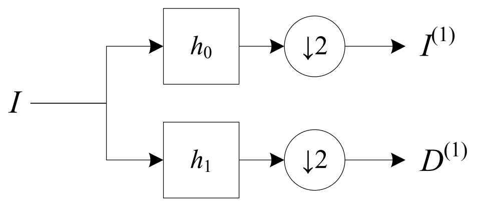


**Figure 2.1** Basic 2-band building block for dyadic DTWT of a subband decomposition tree using the filter pair $\{h_0, h_1\}$ where $h_0$ is low-pass and $h_1$ is high-pass. The filtering operations are followed by a downsampling operation. The next block continues from $I^{(1)}$ which is the low-pass version of the input signal $I(t)$ and $D^{(1)}$ is the detailed subband signal.

The subimages for level $m$ can be written similar to Equation (2.32) as

$$D^{(s,m)}(\mathbf{x}) = \sum_{\mathbf{k}} I(\mathbf{k})\psi^{(s,m)}(2^m \mathbf{x} - \mathbf{k}) \tag{2.35}$$

$$I^{(m)}(\mathbf{x}) = \sum_{\mathbf{k}} I(\mathbf{k})\phi^{(m)}(2^m \mathbf{x} - \mathbf{k}) \tag{2.36}$$

where $\psi^{(s,m)}$ is the **wavelet filter** for subband $s$ and level $m$ and $\phi^{(m)}$ is the **scaling filter** for level $m$. Here, $\mathbf{x} = [x_1, x_2]^T$ refers to the *spatial coordinates* where $x_1$ is the *vertical* and $x_2$ the *horizontal* component. Positive directions are down and to the right. Since these filters are separable, they can be written as products of 1-D wavelet and scaling filters as follows:

$$\begin{aligned}
\psi^{(1,m)}(\mathbf{x}) &= \psi^{(m)}(x_1)\phi^{(m)}(x_2) \\
\psi^{(2,m)}(\mathbf{x}) &= \phi^{(m)}(x_1)\psi^{(m)}(x_2) \\
\psi^{(3,m)}(\mathbf{x}) &= \psi^{(m)}(x_1)\psi^{(m)}(x_2) \\
\phi^{(m)}(\mathbf{x}) &= \phi^{(m)}(x_1)\phi^{(m)}(x_2)
\end{aligned} \tag{2.37}$$

[1] As a common convention, from here on, the DWT will be used to refer to the discrete space or time signals instead of DTWT.

The corresponding partitioning of the unit frequency cell is shown in Figure 2.2. Since the transform is separable, the regions are rectangular.

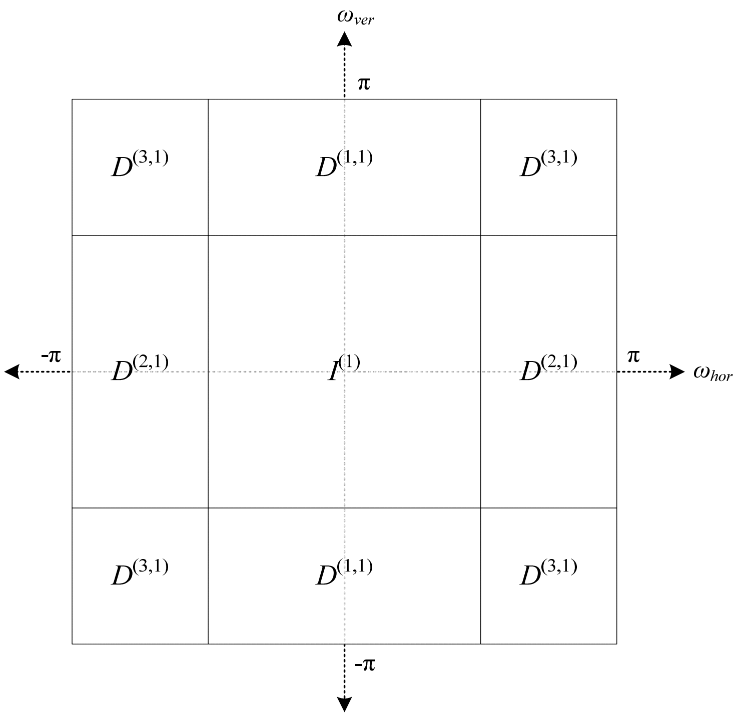


**Figure 2.2** Tiling of the 2-D unit frequency cell by the Discrete Wavelet Transform (DWT). Only the first level is shown. The subsequent levels tile further the low-pass subimage $I^{(1)}$.

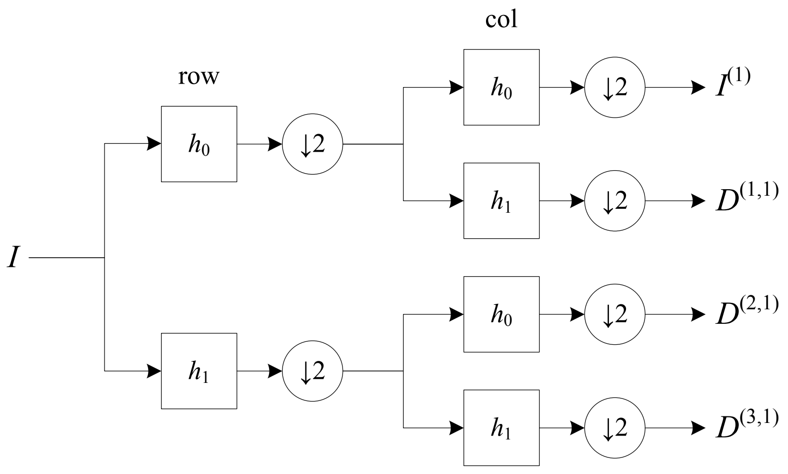


**Figure 2.3** Building block for separable 2-D DWT for images. The same filter pair $\{h_0, h_1\}$ as in the 1-D DWT is used first for the rows than for the columns. The next block continues from the lowpass subimage $I^{(1)}$. Here, $D^{(s,1)}$'s are the detailed subimages for level 1.

### 2.3.3 Wavelets for Motion Estimation

The Discrete Wavelet Transform (DWT) is successfully used for image analysis; however, due to its limitations it is not adequate for motion estimation and video processing. The two major disadvantages of the DWT are shift variance and poor directional selectivity [19][46][1][17][25][31][35][70][77][82][90]. It is shift variant because the transform coefficients behave unpredictably under shifts of the input signal. This is because the DWT is critically sampled, i.e. *N* coefficients are produced for *N* pixels. The two-dimensional DWT has poor directional selectivity because it cannot distinguish between the two diagonal orientations: +45° and -45°. It can only distinguish between three different spatial orientations, namely, the horizontal, vertical and diagonal. For motion estimation, however, these two features are crucial.

The shift-dependency of the discrete wavelet transform hinders its use for motion estimation. Therefore, several modifications to the DWT have been proposed to make the transform suitable for motion-estimation. Most of the modifications are in the way to make the transform shift-invariant. One of these methods is the redundant, or, overcomplete wavelet transform. The redundant DWT (RDWT) removes the downsampling operation from the traditional DWT to ensure shift invariance at the cost of a redundant, or overcomplete representation [60]. The RDWT is also called the “undecimated wavelet transform”.

Cui *et al* [25][26] demonstrated that the redundant (overcomplete) wavelet domain facilitates the placement of an irregular triangular mesh to video images to implement geometries for motion estimation. In addition, they present a new form of multihypothesis prediction, namely the redundant wavelet multihypothesis.

DeVore *et al* [28] also presents another redundant wavelet transform, namely the Redundant Haar Transform. DeVore also claims that it is highly computationally efficient.

Fernandes *et al* [32] proposed a complex, directional double-density wavelet transform (CDDWT) that is almost shift-invariant with redundancy of 2.67 for two-dimensions. They based their transform on the double density wavelet transform

(DDWT) proposed by Selesnick. The CDDWT is created by performing a non-redundant post projection on the double density wavelet transform (DDWT) coefficients.

Wu *et al* [87][88] proposed an iterative motion estimation algorithm that uses wavelet approximation as an optical flow model. The wavelet motion model, originally presented by Cai and Wang, represents motion vectors by a linear combination of hierarchical basis functions. The general idea is to recover the associated wavelet coefficients from the coarsest level to the current level by minimizing the SSD, then reconstruct motion vectors via interpolation and use them to warp the first image toward the second. The wavelet basis functions play triple roles in the algorithm, to construct motion vectors from coarse-to-fine, select multiple large-to-small regions for image matching and impose smoothness. The results are robust and accurate.

Sebe *et al* [77] proposes the Overcomplete Discrete Wavelet Transform (ODWT) to overcome the shift-variant property of the DWT. They demonstrated the possibility to obtain for a given function *f* in the wavelet space, the displaced function *f* with any integer value of the sampling period. They state that it is always possible to obtain any translated version of the original discrete function from one of the ODWT members.

Bernard [8][9] describes a new way to compute the optical flow, based on a discrete wavelet basis analysis. The optical flow is estimated locally by the projection of the differential optic flow equation onto wavelets. The approach has low complexity and permits the measurement of illumination changes, making the optical flow computation more robust.

Cai and Adjouadi [17][18] proposes an efficient approach, called “level-refined motion estimation and subband compensation” (LRSC) method for fast motion estimation and compensation in the wavelet domain. Wavelet domain motion estimation is processed by searching the best matching block between the current subframe and the corresponding reference subframe, from coarse to fine resolutions hierarchically. They apply DWT on the current frame and the reference frame before motion estimations between the subframe pyramids.

Li [53] generalized Lucas and Kanade's scheme [57] into the wavelet space. He discovered that odd-phased and even-phased coefficients of the DWT can be recovered from each other. In this way, the negative artifact of the downsampling operation could be overcome. Thus, he proposed an iterative phase estimation scheme in the wavelet space for the estimation of motion. He constrains its solution space to block-based translational motion models.

Park and Kim [70] proposed the "low-band-shift" (LBS) method to overcome the shift-variant property of the discrete wavelet transform and presented a motion estimation and compensation method in the wavelet domain. In this method, the reference frame is shifted by one pixel along the x, the y, and the diagonal directions, respectively, in the spatial domain. The shifted frames are transformed to the wavelet domain for motion estimation. This process is then repeated iteratively in the next level and so on. The motion vector is then found by selecting the block who gives the minimum mean absolute difference (MAD) between the current wavelet block and the reference wavelet block. Fu *et al* [35] modified this method in order to improve its performance around image boundaries and proposed the "Symmetric Padding Low-Band-Shift Motion Estimation" (LBSME-SP) algorithm. Yuan and Mandal [90] also based their work on LBS and proposed a multi-hypothesis hierarchical backward motion estimation and compensation framework.

Auwera *et al* [2] proposed a block based motion estimation and compensation technique applied on the detail images of the wavelet pyramidal decomposition. The algorithm performs full-search motion estimation on every level of the wavelet decomposition and extracts integer shifts of each block. Methods changes depending on whether the shift is odd or even.

Tjoa *et al* [82] proposed a motion compensation (MC) method in the DWT domain to realize a hierarchical MC. They applied a locally adaptive interpolation filter to dissolve the shift-invariant problem caused by down sampling in DWT. The filter predicts DWT coefficients in the current frame from the coarsely motion compensated DWT coefficients in the reference frame, using the motion vectors estimated from the lower resolution image.

Ates [1] proposed a modified block matching algorithm based on the idea of representing motion information at different resolution levels which are obtained by using biorthogonal wavelet filter pairs.

Xiong *et al.* [89] presents a new approach to deal with the translation- and scale-invariant problem of discrete wavelet transform. They base their approach on the theory of interpolation in the wavelet subspace and adaptively renormalize the original signal by using an orthonormal scale function and the first two moments of the signal. The renormalized signal is then decomposed according to conventional wavelet decomposition, and the final wavelet coefficients are proved to be both translation- and scale-invariant. They use their approach in texture identification.

Mujica *et al.* [66] presents a novel motion parameter estimation algorithm based on the spatio-temporal continuous wavelet transform (CWT). They use the CWT to map the input signal space to a physically meaningful parameter space, which allows for the accurate estimation of motion parameters.

Şendur and Güleryüz [37][38] proposed new interscale edge and occlusion models that are incorporated in a wavelet-based probabilistic motion estimation framework. They formulated the estimation of the motion field as a maximum likelihood optimization problem that allows the use of dynamic programming algorithm in order to find the globally optimum solution.

To summarize, wavelets are being used increasingly for the estimation of motion. Since the DWT is not directly suitable for motion estimation, several methods have been developed to make them suitable for motion estimation. One of such modifications which we preferred in this work is the Complex Discrete Wavelet Transform developed by Magarey and Kingsbury, which will be covered in detail in the next chapter.

# CHAPTER 3

# THE COMPLEX DISCRETE WAVELET TRANSFORM

## 3.1 Introduction

The wavelet transform is a suitable tool for analyzing images. However, for video processing, where consecutive frames have only slight variations of each other, the wavelet transform can not be used effectively, since the wavelet transforms of these frames would be highly different from each other. This is because the wavelet transform is not shift-invariant like the Fourier Transform. Its sensitivity to signal position makes the analysis of video frames a difficult task.

In 1997, Magarey and Kingsbury [60] have developed the Complex Discrete Wavelet Transform (CDWT) in order to efficiently implement the accurate phase-based motion estimation method of Fleet and Jepson [30]. Motion estimation would only be possible if a correspondence on the wavelet coefficients could be established for the moving pixels, which, in its simplest case requires the wavelet transform to be shift invariant. This is the basic principle behind the CDWT. The fact that DWT is critically sampled, which is the result of the downsampling operation performed at each level, causes the transform to be shift-variant. The Continuous Wavelet Transform, on the other hand, does not suffer from this artifact as there is no downsampling operation.

The CDWT is a compact, multiresolutional, orientation-tuned decomposition of an image. The superiority of this method lies in the fact that it combines the shifting property of the Fourier Transform with the scale-independence and localization features of the Wavelet Transform.

The most important features of the CDWT can be summarized as: (a) approximate shift-invariance; (b) good directional selectivity; (c) sub-pixel accuracy; (d) limited redundancy; and (e) Gabor-like complex-valued filters.

The next section briefly describes the CDWT of Magarey. A detailed description can be found in [59]. At the end of this chapter, a presentation of the properties of the CDWT required for the development of the Complex Wavelet Tracker is given.

## 3.2 The Complex Discrete Wavelet Transform

The structure of the 2-D Complex Discrete Wavelet Transform (CDWT) is given in Figure 3.1. It has a similar structure to the 2-D Discrete Wavelet Transform (DWT) with two major differences: (a) the basis filters $\{h_0, h_1\}$ are complex-valued, and, (b) there is a *mirror* branch for the 2-D CDWT to cover the second quadrant of the unit frequency cell (See below).

The CDWT is based on a pair of complex-valued even-length FIR filters with approximate Gabor form [60]

$$h_0(x) = a_0 e^{-\frac{(x-x_0)^2}{2\sigma_0^2}} e^{j\omega_0(x-x_0)} \tag{3.1}$$

$$h_1(x) = a_1 e^{-\frac{(x-x_1)^2}{2\sigma_1^2}} e^{j\omega_1(x-x_1)} \tag{3.2}$$

$$\text{for} \quad x = -L, \ldots, (L-1) \tag{3.3}$$

with $x_0$ set to $-\frac{1}{2}$ to position the Gaussian window symmetrically in the interval $[-L, L-1]$. As in the DWT (See Section 2.3.2), $h_0$ is the lowpass and $h_1$ is the highpass filter. Here, $a_0$ and $a_1$ are *amplitudes*, $\omega_0$ and $\omega_1$ are *modulation frequencies*, $\sigma_0$ and $\sigma_1$ are *window standard deviations*, and $L$ represents the *half-length* of the window.

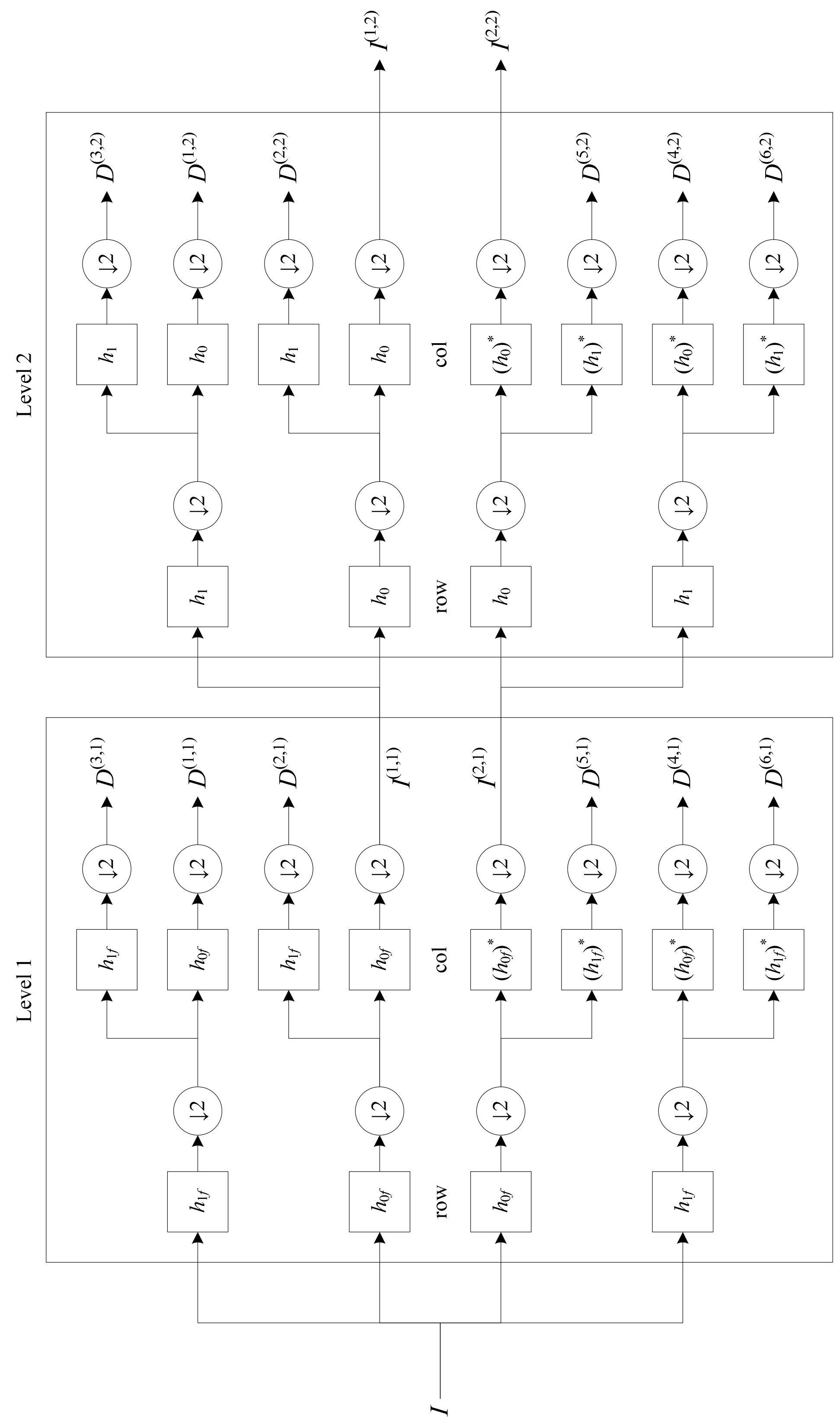


**Figure 3.1** The hierarchical structure of the Complex Discrete Wavelet Transform (CDWT)

The CDWT is implemented using these filters in the standard subband decomposition tree [60] as the DWT. Since $h_0$ and $h_1$ are Gabor-like, the *wavelet* and *scaling* filters, $\psi^{(m)}$ and $\phi^{(m)}$, are designed to be Gabor-like with even length as well.

$$\psi^{(m)}(x) \approx a_m e^{-\frac{(x-x_m)^2}{2\sigma_m^2}} e^{j\omega_m(x-x_m)} \tag{3.4}$$

$$\phi^{(m)}(x) \approx \hat{a}_m e^{-\frac{(x-x_m)^2}{2\hat{\sigma}_m^2}} e^{j\hat{\omega}_m(x-x_m)} \tag{3.5}$$

$$\text{for } \; x = -(2^m - 1)L, \ldots, (2^m - 1)(L-1) \tag{3.6}$$

The parameters of the wavelet and scaling filters are computable from $h_0$ and $h_1$ [60]. Here, $a_m$ and $\hat{a}_m$ are *amplitudes*, $\omega_m$ and $\hat{\omega}_m$ are *modulation frequencies*, $\sigma_m$ and $\hat{\sigma}_m$ are *window standard deviations* of the wavelet and scaling filters, respectively.

The 2-D CDWT is a separable transform whose filters are products of 1-D wavelet and scaling filters [59]. The wavelet filters for the top branch of Figure 3.1 are

$$\psi^{(1,m)}(\mathbf{x}) = \psi^{(m)}(x_1)\phi^{(m)}(x_2) \tag{3.7}$$

$$\psi^{(2,m)}(\mathbf{x}) = \phi^{(m)}(x_1)\psi^{(m)}(x_2) \tag{3.8}$$

$$\psi^{(3,m)}(\mathbf{x}) = \psi^{(m)}(x_1)\psi^{(m)}(x_2) \tag{3.9}$$

and the filters for the bottom (*mirror*) branch are

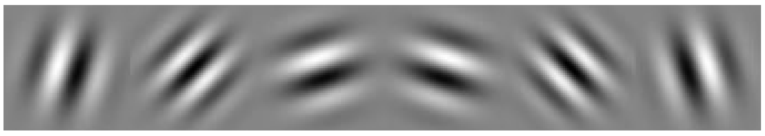
$$\psi^{(4,m)}(\mathbf{x}) = \psi^{(m)}(x_1)\phi^{*(m)}(x_2) \tag{3.10}$$

$$\psi^{(5,m)}(\mathbf{x}) = \phi^{(m)}(x_1)\psi^{*(m)}(x_2) \tag{3.11}$$

$$\psi^{(6,m)}(\mathbf{x}) = \psi^{(m)}(x_1)\psi^{*(m)}(x_2) \tag{3.12}$$

Each of these wavelet filters emphasizes edges on one direction. Thus, six directions can be handled compared to the DWT which can handle only horizontal, vertical and diagonal directions with no distinction among the two diagonals. A grayscale plot of the impulses responses of the wavelet filters for level 4 is shown in Figure 3.2 with gray representing 0.

(a)



(b)

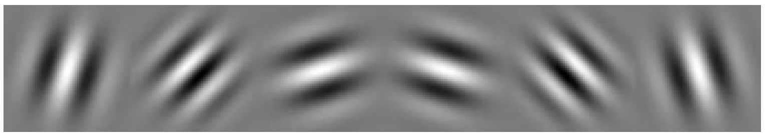


**Figure 3.2** Impulse responses of the six wavelet filters $\psi^{(s,4)}$ for level 4 of the 2-D CDWT. The filters are the rotation-invariant filter pair defined in Equation (3.18) and (3.19). (a) Real part. (b) Imaginary part. From left to right, the order of orientations are $s$ = 2, 3, 1, 4, 6, 5. (This Figure is reproduced from [59].)

The 2-D CDWT equivalent wavelet filters can also be written as 2-D Gabor filters [59]:

$$\psi^{(s,m)}(\mathbf{x}) = a^{(s,m)} N(\mathbf{x} \mid \mathbf{x}_m, \Lambda_{s,m}) e^{j\left(\Omega^{(s,m)}\right)^T (\mathbf{x}-\mathbf{x}_m)} \tag{3.13}$$

where $N(\mathbf{x}|\mathbf{x}_m,\Lambda)$ is an unnormalized bivariate Gaussian in $\mathbf{x}$ with mean $\mathbf{x}_m$ and covariance $\Lambda$:

$$N(\mathbf{x} \mid \mathbf{x}_m, \Lambda) = \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}_m)^T \Lambda^{-1} (\mathbf{x}-\mathbf{x}_m)\right] \tag{3.14}$$

Because the transform is separable, the covariance matrix $\Lambda_{s,m}$ is always diagonal [59] with entries either $\sigma_m^2$ or $\hat{\sigma}_m^2$ . Here, $\Omega^{(s,m)}$ represents the center spatial frequency of subband $s$ of level $m$. Referring to Equations (3.7) through (3.12) we can write

$$\begin{aligned}
\Omega^{(1,m)} &= [\omega_m, \hat{\omega}_m]^T \\
\Omega^{(2,m)} &= [\hat{\omega}_m, \omega_m]^T \\
\Omega^{(3,m)} &= [\omega_m, \omega_m]^T \\
\Omega^{(4,m)} &= [\omega_m, -\hat{\omega}_m]^T \\
\Omega^{(5,m)} &= [\hat{\omega}_m, -\omega_m]^T \\
\Omega^{(6,m)} &= [\omega_m, -\omega_m]^T
\end{aligned} \tag{3.15}$$

The center frequency $\Omega^{(s,m)}$ of each filter specifies its direction of constant phase. These orientations are shown in Figure 3.2.

The 1-D wavelet and scaling filters are Hardy-like filters [59], i.e. they eliminate negative frequencies. This is shown in Figure 3.3 for the first 4 levels. Therefore, for 2-D, only the first quadrant of the unit frequency cell is covered. However, for real images, the second quadrant is needed as well [59]. In order to cover the second quadrant, a mirror branch is implemented using the complex conjugates of the $h_0$ and $h_1$ filters for the columns. As a result, the overall redundancy of the 2-D CDWT is 4:1. 2:1 because it is complex valued and another 2:1 due to the mirror branch, resulting in two low-pass and six detailed complex subimages at each level.

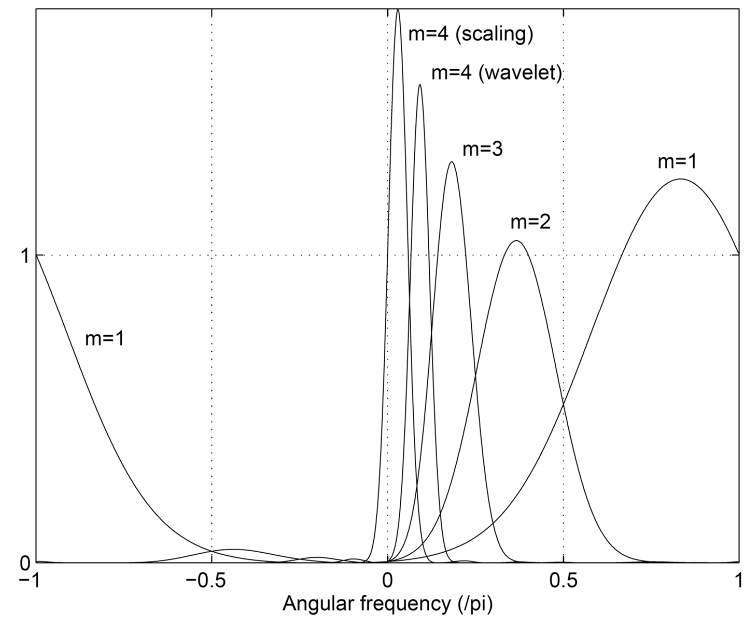


**Figure 3.3** Magnitude frequency responses of CDWT wavelet ($m$ = 1, 2, 3, 4) and scaling filter ($m$ = 4), using $\{h_0, h_1\}$ with parameters: $a_0 = 0.39$, $a_1 = 0.39j$, $\omega_0 = \pi/6$, $\omega_1 = 5\pi/6$, $\sigma_0 = 1.27$, $\sigma_1 = 1.27$, and $L = 4$. Note that the negative frequencies are approximately neglected for $m > 1$. (This Figure is reproduced from [59].)

A *prefilter f* is used for the first level filters in order to form a perfectly scaled wavelet decomposition [59]. Then, the first level filters become

$$h_{0f} = h_0 * f \tag{3.16}$$

$$h_{1f} = h_1 * f \tag{3.17}$$

The *f*-modified wavelet and scaling filters are still in Gabor form [59]. The effect of the prefilter is that it scales the maximum amplitudes of the wavelet and scaling filters shown in Figure 3.3 to the same level.

The approximate partitioning of the frequency is shown in Figure 3.4 for the first level. In contrast to DWT, the partitions are elliptical centered on $\Omega^{(s,m)}$.

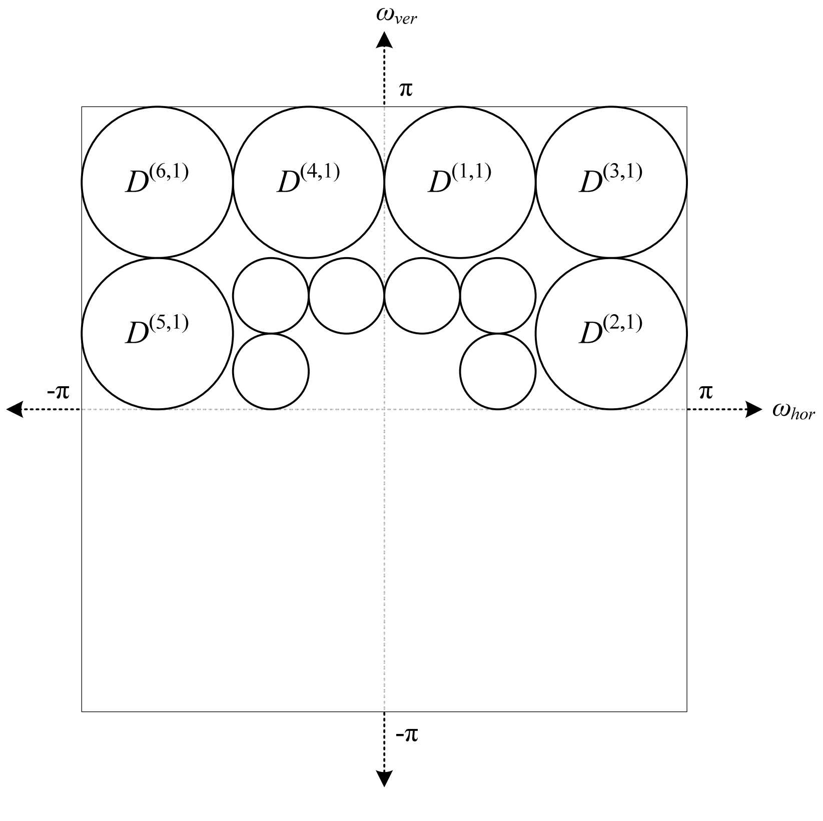


**Figure 3.4** Tiling of the 2-D unit frequency cell by the Complex Discrete Wavelet Transform (CDWT). Only the first two levels are shown. The inner circles are for the second level. The subsequent levels tile further the low-pass subimages $I^{(1,m)}$ and $I^{(2,m)}$ in the same manner. The subimages on the left hand side are produced by the mirror filters.

The following *rotation-invariant* (RI) filter pair defined in [59] is used for this work:

$$h_0 = [(1-j)\ \ (4-j)\ \ (4+j)\ \ (1+j)]\ /\ 10 \tag{3.18}$$

$$h_1 = [(-1-2j)\ \ (5+2j)\ \ (-5+2j)\ \ (1-2j)]\ /\ 14 \tag{3.19}$$

The corresponding prefilter for perfect scaling is given by

$$f = [-j \quad 5 \quad j] \ / \ 5 \tag{3.20}$$

This filter pair can be approximated with the following Gabor parameters $a_0 = 0.47$, $a_1 = 0.43j$, $\omega_0 = \pi / 6$, $\omega_1 = 0.76\pi$, $\sigma_0 = 0.97$, $\sigma_1 = 1.07$, and $L = 2$ using Equations (3.1) and (3.2). The RI filter pair has several advantages. (a) It has a length of 4 which is which quite low and important in decreasing computational cost. (b) The coefficients of $h_0$, $h_1$, and $f$ are rational-valued making a fixed-point implementation possible. (c) The wavelet filters are Hardy-like, i.e. they neglect negative frequencies. (d) The frequency cell tiling is optimal and independent of $m$. (e) the orientations of the filters are 19°, 45°, 71°, 109°, 135°, and 161° which is very close to optimum spacing.

## 3.3 Properties of the CDWT

The CDWT has several properties which makes it suitable for motion estimation and tracking. This Section reviews some of these properties as required in the context of this thesis.

### 3.3.1 Shiftability and Coefficient Interpolation

The feature that makes CDWT based motion estimation possible is the approximate shift-invariance feature. This feature is accomplished by the *shiftability* property of the CDWT. The shiftability is defined by Simoncelli *et al.* [80] and used by Magarey [59][60]. A transform is said to be *shiftable* if the transform of the shifted input is computable as a weighted sum over the original transform coefficients. This applies to CDWT as

$$D^{(s,m)}(\mathbf{x}+\mathbf{f}) \approx \sum_{\mathbf{k}} W_{\mathbf{f}}^{(s,m)}(\mathbf{k}) D^{(s,m)}(\mathbf{x}+\mathbf{k}) \tag{3.21}$$

where $W_{\mathbf{f}}^{(s,m)}$ is a lowpass kernel modulated to the center frequency of the equivalent wavelet filter:

$$W_{\mathbf{f}}^{(s,m)}(\mathbf{k}) = H_{\mathbf{f}}(-\mathbf{k}) e^{j2^m \left(\Omega^{(s,m)}\right)^T (\mathbf{f}-\mathbf{k})} \tag{3.22}$$

and $H_\mathbf{f}$ is a lowpass kernel, which can be separable given that the transform is separable:

$$H_\mathbf{f}(\mathbf{k}) = h_{f_1}(k_1)h_{f2}(k_2)\,. \tag{3.23}$$

Magarey [59] showed that the key to the interpolability of the wavelet coefficients is the interpolability of its equivalent wavelet filter after downsampling. Real-valued critically-sampled transforms cannot be shiftable in this sense.

Among several types of interpolators suggested by Magarey in [59] two of which are of interest for us: The *windowed sinc* function and the *staircase* interpolator.

### 3.3.1.1 Windowed Sinc Interpolator

The four-tap windowed sinc function is defined as [59]

$$h_f(k) = \begin{cases} g_f(k)\dfrac{\sin\pi(f+k)}{\pi(f+k)} & f+k \neq 0 \\ 1 & f+k = 0 \end{cases} \tag{3.24}$$

where the *windowing* function $g_f(k)$ is given by

$$g_f(k) = \begin{cases} \dfrac{\cos\frac{\pi}{2}(f+k)}{1-(f+k)^2} & |f+k| \neq 1 \\ \dfrac{\pi}{4} & |f+k| = 1 \end{cases} \tag{3.25}$$

for $k = -2, \ldots, 1$. This kernel is used in the coefficient interpolation phase during the coarse to fine approach.

### 3.3.1.2 Staircase Interpolator

The staircase interpolator is the Lagrange interpolator of length 1, which is defined as

$$h_f(k) = \begin{cases} 1 & \mathit{if}\ k = 0 \\ 0 & \mathit{otherwise} \end{cases} \tag{3.26}$$

The CDWT subband interpolation formula (3.21), using the staircase interpolator, will then be

$$D^{(s,m)}(\mathbf{x}+\mathbf{f}) \approx D^{(s,m)}(\mathbf{x})e^{j\theta^{(s,m)}(\mathbf{f})} \tag{3.27}$$

where

$$\theta^{(s,m)}(\mathbf{f}) = 2^m \left(\Omega^{(s,m)}\right)^T \mathbf{f} \tag{3.28}$$

for

$$\mathbf{f} \in [-0.5, 0.5]\times[-0.5, 0.5]. \tag{3.29}$$

Equation (3.27) gives the relation between input shifts and CDWT behavior. It shows that the CDWT phase changes linearly with shifts in the input where the magnitude remains approximately the same. This is the key for the motion estimation algorithm developed by Magarey. This linear relationship provides a model for the phase behavior around an integer-indexed CDWT subband coefficient as a *plane* whose gradient is the center frequency of the associated wavelet filter, scaled up by $2^m$.

### 3.3.2 Energy of Wavelet Filters

The energy of the wavelet filter $\psi^{(s,m)}$ is denoted as $P^{(s,m)}$ and defined by [59]

$$P^{(s,m)} = \frac{1}{(2\pi)^2} \int_0^{2\pi}\int_0^{2\pi} \left|\Psi^{(s,m)}(\Omega)\right|^2 d\Omega \tag{3.30}$$

It is shown in [59] that the only $m$-dependence of $P^{(s,m)}$ is a factor of $(4\lambda^4)^m$, where, $\lambda$ is a scaling factor obtained by computing the prefilter $f$ which is equal to $|H_0(0)| / 2$.

### 3.3.3 CDWT Subband Noise

Assuming that the original frame noise is Gaussian then the CDWT subband noise $e^{(s,m)}$ in subband $s$ of level $m$ is also Gaussian. It is shown in [59] that the CDWT subband noise pdf becomes

$$p\left(e^{(s,m)}\right) \propto \exp\left\{-\frac{\left|e^{(s,m)}\right|^2}{2\sigma_{s,m}^2}\right\} \tag{3.31}$$

where

$$\sigma_{s,m}^2 = \sigma^2 P^{(s,m)} \tag{3.32}$$

Here, $\sigma^2$ denotes the original frame noise variance and $P^{(s,m)}$ denotes the *energy* of the wavelet filter $\psi^{(s,m)}$, which is given in Equation (3.30) (See Section 3.3.2).

## 3.4 Summary

The Complex Discrete Wavelet Transform (CDWT) developed by Magarey and Kingsbury is presented. The CDWT uses a complex-valued Gabor-like basis filter pair. The filters are approximate Hardy-like, which neglect the negative frequencies. For images, this causes only the first quadrant of the unit frequency cell to be covered. In order to include the second quadrant as well, a mirror filtering branch is used. A prefilter is applied to convert the transform to a perfectly scaled structure. At each levels six detailed and two lowpass subimages are produced. The redundancy of the CDWT is four transform coefficients for one pixel.

The shiftability property is introduced which is the key in the interpolation of the CDWT coefficients, thus, given the CDWT its approximate shift-invariance property. The energy of the wavelet filter and the CDWT subband noise properties are presented at last.

# CHAPTER 4

# THE COMPLEX WAVELET TRACKER

## 4.1 Introduction

The Complex Wavelet Tracker is a vision based tracking algorithm that uses a parametric motion model to adapt the track gate to the motion of the target throughout the image sequence. The parametric motion model allows modeling the expected deformations of the target resulting on the 2-D image plane. The algorithm operates in the complex discrete wavelet domain and uses the Complex Discrete Wavelet Transform (CDWT) developed by Magarey [60].

The CDWT is based on the local translational motion model [60]. Therefore, Castellano [19] proposed to modify the CDWT in order to accommodate for affine deformations. In her work, the effect of an affine transformation applied in the spatial domain on the corresponding wavelet domain has been analyzed. The initial steps for deriving an orientation-shiftable CDWT were shown. However, it was mentioned that the design of an orientation-shiftable CDWT even for *semi*-affine motion estimation would not be a trivial task.

The CDWT is an elegant and compact transformation that allows for estimating *true* motion vectors [60] in a very precise and robust manner. Taking into account this fact, we explored several alternatives to adapt this powerful tool in the area of tracking. Instead of modifying the CDWT itself to accommodate for affine deformations, we explored the possibility of using the CDWT as is and reformulated the motion estimation method accordingly. So, we based our approach on the CDWT based motion estimation method developed by Magarey [60].

The CDWT based motion estimation (CDWT-ME) algorithm is developed with an analogy to the block matching based approach [59]. Magarey has stated the differences between these two approaches. The most important difference for the context of this thesis is the fact that estimation in the CDWT domain using matching at only *one* pixel is possible rather than over a region of pixels as in block matching schemes. This is the key for the success of the CDWT-ME algorithm in producing a robust and dense optical flow field.

This unique property allows for deriving motion parameters at a direct, single step, without the need for iterations. The displacement for each pixel is computed from the minimum of a quadratic surface defined over the CDWT subbands. Hence, the surfaces for each individual target pixel can be combined to obtain a unique solution for the track parameters of the target. In this way, the resulting motion will be the *true* motion of the target pixels. This is not the case when applied in the spatial domain. For Lucas-Kanade [57], Hager and Belhumeur [39] and Baker-Matthews [3], iteration is required to converge to a solution.

Another point that is stated by Magarey is that the confidence of the CDWT-ME method is reduced if the contributing region undergoes significant *affine* motion, as the method was developed for estimating translational motion. However, extensive simulations on the CDWT-ME method showed that the dense flow field is quite successful even under *affine* deformations of the input frames. These motion estimation results are given in Section 6.1.

The theoretical estimation range for the CDWT-ME method is ±0.5 pels for a *subpel*[2], which results in a total of $\pm 0.5 \times 2^m$ pixels for the original frame. This will be equal to ±8 pixels for a four level pyramid. This is quite a good value for tracking, since, the resulting target motion between successive frames captured at 25 Hz or 30 Hz is quite limited. Hence, the theoretical limits of maximum *affine* deformation for a region can be obtained using the limits for pixel displacement. The next Section captures these limits and the results are given in Section 6.2.3.

[2] According to Magarey's use a *subpel* refers to a pixel location at a lower resolution ($m > 0$). Thus, subpel at level $m$ corresponds a block of $2^m \times 2^m$ pixels in the original image.

The high-level structure of the Complex Wavelet Tracker (CWT) is shown in Figure 4.1. Frames are first entering the gate adjustment block where they are clipped according to the track gate. Then the CDWT is of the frames are computed. Finally, the track parameters are estimated using the last two frames.

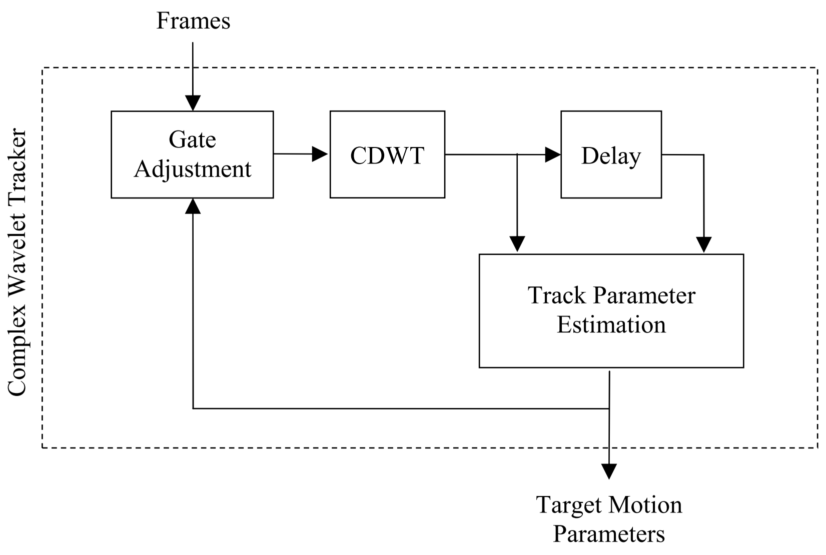


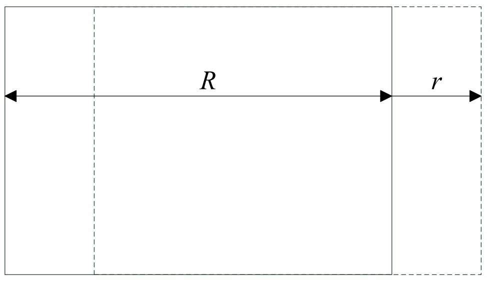
**Figure 4.1** The high-level structure of the Complex Wavelet Tracker. The CDWT of the input frames are used in computing the target motion parameters.

The next Section explores the theoretical limits of deformation that can be handled by the CDWT. This also serves as the justification of our approach. This Section is then followed by the definition of the tracking problem for the purpose of this thesis. After the introduction of parametric motion models, the structure of the CWT algorithm is presented. The formulation of the tracking algorithm starts with Section 4.6. This is presented in two sections. First, single level estimation of the track parameters is described, which is followed by the extension to hierarchical tracking.

## 4.2 CDWT Limits of Deformation

In order to have an overall feeling of the deformation handling capacity of the CDWT, the theoretical deformation limits for a 4-level CDWT pyramid will be investigated. An *affine* deformation includes translation, scaling, rotation and shear type of motion. A four level CDWT allows for a ±8 pixel maximum pixel displacement.

For **translation**, this limit corresponds to the amount of motion relative to the size of the object. Figure 4.2 depicts this case. The translation is limited to the ratio of the size of the object to the amount of the displacement. For a maximum displacement of 8 pixels for an object of size 64 pixels, the amount of translation will be 12.5%.

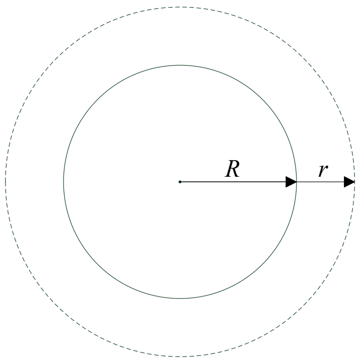


Maximum Translation

$$tr < \frac{r}{R}$$

**Figure 4.2** Maximum translation allowed for an object of size *R* in the direction of motion.

For **scaling**, this limit corresponds to the amount of change in the size of an object relative its original size. Figure 4.3 shows this case. The amount of scaling is limited by the motion of the pixels at the outer boundary of the object. For a maximum displacement of 8 pixels for an object of size 64 pixels, the maximum amount of scaling will be 25%.

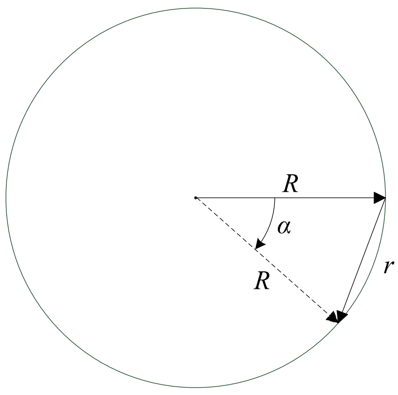


Maximum Scaling

$$sc < \frac{r}{R}$$

**Figure 4.3** Maximum scaling allowed for an object of size 2*R*.

For **rotation**, this limit corresponds to the amount of rotation angle with respect to the size of the object. Figure 4.4 shows this case. The amount of rotation is limited to the maximum displacement of the pixels at the outer region of the object. For a maximum displacement of 8 pixels for an object of size 64 pixels, the maximum amount of rotation will be 14°.

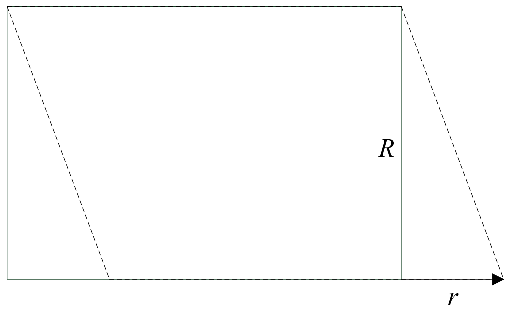


Maximum Rotation

$$\alpha < 2\arcsin\left(\frac{r}{2R}\right)$$

**Figure 4.4** Maximum rotation allowed for an object of size $2R$.

For **shear**, this limit corresponds to the amount of motion relative to the size of the object. Figure 4.5 depicts this case. The shear is limited to the maximum displacement of the pixels at the outer edge of the object. For a maximum displacement of 8 pixels for an object of size 64 pixels, the amount of shear will be 12.5%.



Maximum Shear

$$sh < \frac{r}{R}$$

**Figure 4.5** Maximum shear allowed for an object of size $R$ in the direction of motion perpendicular to the perpendicular edge.

A summary of the theoretical limits for representative object sizes are given in Table 4.1. Although practical limits might be less than these values the amounts are quite promising.

**Table 4.1** Amount of deformation allowed for representative object sizes when a 4-level CDWT pyramid is used, which theoretically corresponds to a maximum displacement of 8 pixels.

| *Object Size* | *Translation* | *Scale* | *Rotation* | *Shear* |
|---|---|---|---|---|
| 32 × 32 pixels | 25% | 50% | 29° | 25% |
| 64 × 64 pixels | 12.5% | 25% | 14° | 12.5% |
| 160 × 160 pixels | 5% | 10% | 5° | 5% |

Simulations related with these limits have been performed and the results obtained are presented in Section 6.2.3.

## 4.3 The Tracking Problem

Visual tracking can be defined in many different ways depending on the specific application it is developed for, as there is no *universal* tracking algorithm that can be used for all cases. For our purpose, we prefer the definition of Hager and Belhumeur. In [39], they define tracking equivalent to recovering the motion parameter vector of the target region for each image in the tracking sequence.

We define the target by an enclosing region named the *track gate*. The motion of the track gate is modeled by a *parametric motion model*, i.e. we describe the motion of the track gate in terms of a parametric motion model. We assume that the changes in the track gate within two consecutive frames can be completely described by this motion model. The problem then converges to the estimation of these parameters. Hence, we define t*racking* as the estimation of these parameters throughout the image sequence.

The tracker is initiated by the operator by selecting the target via a rectangular track gate. Throughout the frames, the tracker maintains the track gate on the target. The

CDWT coefficients are used to estimate the motion parameters of the track gate. Figure 4.6 shows samples of the track gate starting from the first frame where it is a rectangle and in frames later on where it has undergone affine changes.

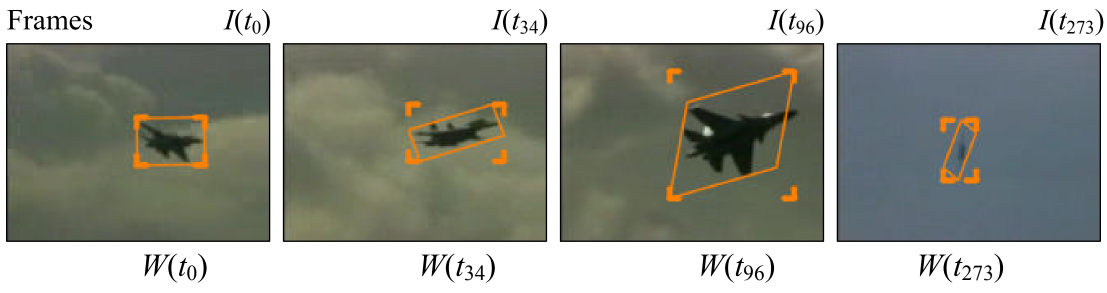


**Figure 4.6** Possible changes in the *track gate* throughout successive frames during tracking. At time $t_0$, the operator selects the target by enclosing it inside a rectangular region. Then, the tracker adjusts the shape of the track gate according to the changes of the target to maintain the gate upon the target.

Let $I(\mathbf{x}, t_n)$ denote the brightness value of the pixel at the location $\mathbf{x} = [x_1, x_2]^T$ on the frame acquired at time $t_n$. Let $W(t_n) = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the set of $N$ target pixels that constitute the target within the track gate for time $t_n$. Over time, the 2-D projected image of the target may translate and deform. The track gate is defined in subpixel accuracy. So, as $W(t_n)$ actually represents the pixels within the track gate of frame $t_n$, the number of pixels $N$ might change as the gate area increases or decreases during the track. This will be denoted by $N(t_n)$, as necessary. Figure 4.6 depicts the case for the changes of the track gate throughout the track.

For simplicity, we define the track gate as a rectangle and use a mask $w(\mathbf{x})$ for $\forall \mathbf{x} \in W(t_n)$ to distinguish between target and background pixels. The track gate is described by the location of its four corners in subpixel accuracy. The mask $w(\mathbf{x})$ is then a weighting function which accompanies each pixel within the track gate.

We model the motion of the track gate by a *parametric motion model* $\mathbf{M}(\mathbf{x}; \mathbf{p})$ parameterized by $\mathbf{p} = [p_1, \ldots, p_k,]^T$. We call $\mathbf{p}$ as the *track parameter vector* where $k$ is the number of parameters used in the motion model. The parameters are estimated on a frame by frame basis, so a time index will be used to distinguish among the parameter vectors, $\mathbf{p}(t_{n-1})$, as required. The function $\mathbf{M}(\mathbf{x}; \mathbf{p})$ takes the pixel at location **x** of the frame $I(\mathbf{x}, t_{n-1})$ and maps it to a subpixel location on the next frame $I(\mathbf{x}, t_n)$ using the parameters $\mathbf{p}(t_{n-1})$. For $\mathbf{p} = \mathbf{0}$, let $\mathbf{M}(\mathbf{x}; \mathbf{0}) = \mathbf{x}$. To obtain a unique solution, it is required that the number of target pixel $N(t_n)$ within the track gate $W(t_n)$ are greater or equal to the number of parameters $k$, namely, $N(t_n) \geq k$. In other words, the size of the track gate should be larger than the number of parameters in order to obtain a unique solution. Assume that **M** is differentiable in both **x** and **p**. Then, recovering the track parameter vector for each frame in the image sequence will be considered equivalent to "tracking the object" [39].

The image sequence will be denoted as $I(t_n)$ where $t_n$ depicts the time at which the frame is acquired. For simplicity we will start the time for the image sequence with $n = 0$ at the frame the target is selected. Then, the first estimation will be performed from frame $I(t_0)$ to frame $I(t_1)$ and the track parameters will be $\mathbf{p}(t_0)$.

## 4.4 Parametric Motion Models

Parametric motion models aim to describe the projection of 3-D motion of a surface into the 2-D image plane [81]. We define the motion model using the warping function **M**. The warp $\mathbf{M}(\mathbf{x};\mathbf{p})$ takes the pixel at location **x** of the frame $I(\mathbf{x}, t_{n-1})$ at and maps it to a subpixel location on the next frame $I(\mathbf{x}, t_n)$. The set of allowed warps depends on the type of motions expected from the object being tracked. Three types of motion models will be presented here: (a) *translational* motion model, (b) *similarity* motion model, and (c) *affine* motion model.

### 4.4.1 Translational Motion Model

The *translational* motion model represents pure displacements of the track gate. If the object is a roughly planar object and its shape remains approximately constant, then the *translational* motion model could be used. This model is defined as

$$\mathbf{M}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} x_1 + p_1 \\ x_2 + p_2 \end{bmatrix}. \tag{4.1}$$

Two parameters are used by this model to define the displacement of the track gate in two directions.

### 4.4.2 Similarity Motion Model

The *similarity* motion model defines a set of similarity warps where the shape of the object remains fixed but its size and orientation changes in addition to location. More specifically, the aspect ratio of the rectangle remains constant. Scaling, rotation and translation are defined. Then, the model will be

$$\mathbf{M}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} (1+p_1)x_1 - p_2x_2 + p_3 \\ p_2x_1 + (1+p_1)x_2 + p_4 \end{bmatrix}. \tag{4.2}$$

Four parameters are required for this type of motion. Note that scaling is equal for both dimensions.

### 4.4.3 Affine Motion Model

The *affine* motion model is used to describe the 2-D motion resulting from a 3-D rigid motion of a planar surface under orthographic projection [81]. This is a more general representation which includes shear type of deformations as well, in addition to translation, scaling and rotation. The *affine* motion model is defined as

$$\mathbf{M}(\mathbf{x};\mathbf{p}) = \begin{bmatrix} (1+p_1)x_1 + p_3x_2 + p_5 \\ p_2x_1 + (1+p_4)x_2 + p_6 \end{bmatrix}. \tag{4.3}$$

This model requires six parameters. In this model, scaling and shear rates can be different for each dimension.

## 4.5 Structure of the Tracking Algorithm

The Complex Wavelet Tracker is a hierarchical algorithm where the estimation is performed on the CDWT coefficients in a hierarchical way from coarse to fine resolution levels. At each level, estimation is performed in the same way and the parameter estimates are propagated to the next finer resolution level where they are used as a starting point. The hierarchical structure of the tracking algorithm is shown in Figure 4.7.

The estimation is performed over two frames: the previous and the current frame observations. Therefore, the track parameter vector **p** gives only the solution from the previous frame to the current frame. Hence, the warping function **M(x;p)** gives the motion of the track gate from frame $I(t_{n-1})$ to frame $I(t_n)$.

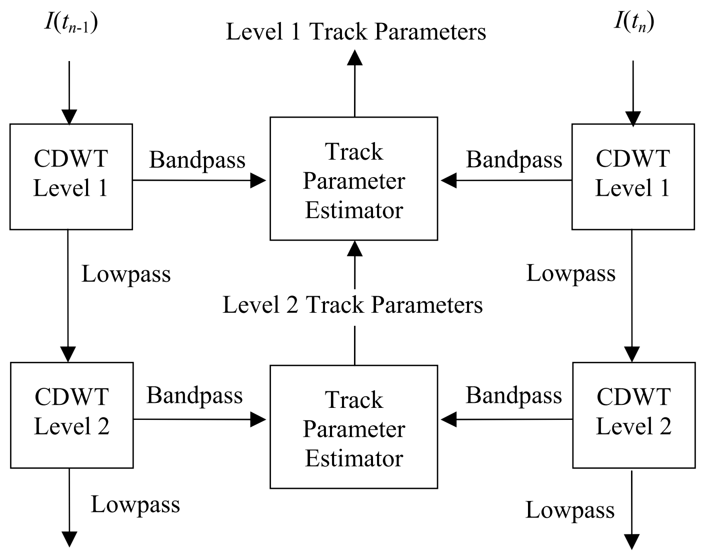


**Figure 4.7** The hierarchical structure of the Complex Wavelet Tracker. Here, only two levels are shown.

The Complex Wavelet Tracker is based on the CDWT-ME algorithm developed by Magarey. The six detailed subbands are used for the estimation of the track parameters. The subbands in the next finer levels are warped according to the estimates of the previous level in order to have a better estimation performance. This is referred as the *refining strategy* in [59]. The steps of the tracking algorithm are as follows:

1. Compute the CDWT's of the two input frames $I(t_{n-1})$ and $I(t_n)$. (See Section 3.2)
2. For the coarsest level ($m = m_{\max}$), compute the track parameters (see Section 4.6.5) and store these parameters as cumulative track parameters.
3. For the next finer level ($m_{\max} > m \geq m_{\min}$):
    a. Scale the cumulative track parameters to the current level $m$. (See Section 4.7.1)

b. Warp $D^{(s,m)}(\mathbf{x}, t_{n-1})$ of this level according to the scaled parameter vector $\mathbf{p}$. (See Section 4.7.2)

c. Compute the track parameters. (See Section 4.6.5)

d. Invert the warping process in step b and obtain the track parameters as if there were no warping. (See Section 4.7.3)

e. Add the track parameters to the previously accumulated parameters to obtain the cumulative track parameters. (See Section 4.7.4)

4. Repeat step 3 until the finest level is reached ($m = m_{\min}$)
5. Scale the cumulative track parameters to the original frame resolution ($m = 0$).

## 4.6 Single Level Track Parameter Estimation

The development of the Complex Wavelet Tracker is based on the formulation of the motion estimation method developed by Magarey and Kingsbury [60]. We will start with a Bayesian formulation of the problem and arrive to an analytical solution.

### 4.6.1 Motion Model

The formulation of Magarey and Kingsbury's CDWT-ME starts with the *local translation model* which may be written as[3]

$$A(\mathbf{x}, t_{n-1}) = A(\mathbf{x} + \mathbf{d}(\mathbf{x}), t_n) \tag{4.4}$$

$$\text{for } \mathbf{x} \in W^{(m)}(\mathbf{x}) \tag{4.5}$$

where $A$ is the *true* underlying image and $A(\mathbf{x}, t_n)$ denotes the *true* brightness value of the pixel at the location $\mathbf{x} = [x_1, x_2]^T$ on the frame at time $t_n$. Here, the window $W^{(m)}(\mathbf{x})$ represents the contributing region of subpel $\mathbf{x}$ of level $m$ on the frame at time $t_{n-1}$. This means that, the motion of the set of pixels at the original frame ($m = 0$) that are contributing to the subpel $\mathbf{x}$ at level $m$ depend on the displacement of that subpel[4]. Here, it is assumed that the only changes in $A$ with time are due to the motion of 2-D

[3] In his work [58][60] Magarey was considering *backward* motion vectors, in tracking, however, *forward* motion is required. Therefore, the original formulation is changed accordingly.

[4] A subpel at level $m$ corresponds to $2^m \times 2^m$ pixels at level 0.

projected intensity patterns. We changed the simple *translational* motion model with a generic parametric motion model which can be written as

$$A(\mathbf{x}, t_{n-1}) = A(\mathbf{M}(\mathbf{x};\mathbf{p}), t_n) \tag{4.6}$$

$$\text{for } \mathbf{x} \in W(t_{n-1}) \tag{4.7}$$

where $W(t_{n-1})$ is the set of pixels within the track gate on the frame acquired at time $t_{n-1}$. Here, $\mathbf{M}(\mathbf{x};\mathbf{p})$ denotes the parameterized set of allowed deformations of the track gate, where $\mathbf{p} = [p_1, \ldots, p_k]^T$ is the *track parameter vector*. The warp $\mathbf{M}(\mathbf{x};\mathbf{p})$ takes the pixel at location $\mathbf{x}$ and moves it to a subpixel location $\mathbf{M}(\mathbf{x};\mathbf{p})$ with respect to the frame coordinates. In order to have a parallel formulation with Magarey we define also the *differential* warping function that gives the displacement vector of the current pixel $\mathbf{x}$ relative to its previous location. The differential warp for each pixel within the track gate is then defined as

$$\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}) = \mathbf{M}(\mathbf{x};\mathbf{p}) - \mathbf{x} \tag{4.8}$$

Before continuing with the formulation of the estimates of $\mathbf{p}$, note the two main differences in this formulation with the one from Magarey. First difference is the increase in the contributing region, and second, the inclusion of the motion model.

### 4.6.2 Bayesian Approach

Our aim is to estimate the track parameter vector $\mathbf{p}$ given two frame observations. Generally, we can only *observe* a frame $I(\mathbf{x}, t)$ that is corrupted by additive noise [81]. So, we assume that the observed image has been acquired by a process in which zero-mean additive white Gaussian noise $e$ with variance $\sigma^2$ corrupts $A$:

$$I(\mathbf{x}, t) = A(\mathbf{x}, t) + e(\mathbf{x}, t)\,. \tag{4.9}$$

Then, applying this to the model in Equation (4.6) we obtain

$$I(\mathbf{x}, t_{n-1}) - e(\mathbf{x}, t_{n-1}) = I(\mathbf{M}(\mathbf{x};\mathbf{p}), t_n) - e(\mathbf{x}, t_n)\,. \tag{4.10}$$

We will first derive an estimate for a single subpel in a single subband, then, using all six subbands we derive the estimate for one subpel in a level. Finally, for our case, we will extend this estimate to cover the set of subpels that are within the track gate.

#### 4.6.2.1 Parameter Estimation for a Single Subband

Since we are interested in the CDWT subbands, we require this relation to be transferred to the complex wavelet domain. As the CDWT is a linear transform [60], we can write Equation (4.10) for subband $s$ of level $m$ as

$$D^{(s,m)}(\mathbf{x},t_{n-1}) - e^{(s,m)}(\mathbf{x},t_{n-1}) = D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(s,m)}),t_n) - e^{(s,m)}(\mathbf{x},t_n) \tag{4.11}$$

$$D^{(s,m)}(\mathbf{x},t_{n-1}) - D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(s,m)}),t_n) = e^{(s,m)}(\mathbf{x}) \tag{4.12}$$

$$\text{for } s = 1, \ldots, 6 \text{ and } \mathbf{x} \in W^{(m)}(t_{n-1}) \tag{4.13}$$

where $D^{(s,m)}(\mathbf{x},t_n)$ denotes the detailed CDWT subband $s$ of level $m$. Here, $\mathbf{p}^{(s,m)}$ denotes the parameter vector that is valid for subband $s$ of level $m$. $W^{(m)}$ represents the set of subpels within the track gate scaled down to level $m$. The differential noise $e^{(s,m)}(\mathbf{x}) = e^{(s,m)}(\mathbf{x},t_{n-1}) - e^{(s,m)}(\mathbf{x},t_n)$ in subband $s$ of level $m$ is the CDWT of the original frame differential noise and is shown to be Gaussian in [59]. The CDWT subband differential noise probability density function is given in Equation (3.31) (See Section 3.3.3).

The goal is to determine the most likely $\mathbf{p}^{(s,m)}$. To obtain the maximum *a posteriori* (MAP) estimate of the track parameters $\mathbf{p}^{(s,m)}$ of $\mathbf{x}$ for subband $s$ of level $m$, the *posterior* distribution of $\mathbf{p}^{(s,m)}$ given the CDWT of the two frame observations are required [50][81]. Then the MAP estimate of the track parameters can be written as

$$\hat{\mathbf{p}}^{(s,m)}(\mathbf{x}) = \arg\max_{\mathbf{p}} \left\{ p\left(\mathbf{p}^{(s,m)} \middle| D^{(s,m)}(\mathbf{x},t_{n-1}), D^{(s,m)}(\mathbf{x},t_n)\right) \right\} \tag{4.14}$$

where $p(\cdot)$ denotes the *probability density function* (pdf). Applying Bayes' rule to the *posterior* pdf, it can be shown that

$$p\left(\mathbf{p}^{(s,m)} \middle| D^{(s,m)}(\mathbf{x},t_{n-1}), D^{(s,m)}(\mathbf{x},t_n)\right) = \frac{p\left(D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(s,m)}, D^{(s,m)}(\mathbf{x},t_{n-1})\right) p\left(\mathbf{p}^{(s,m)} \middle| D^{(s,m)}(\mathbf{x},t_{n-1})\right)}{p\left(D^{(s,m)}(\mathbf{x},t_n) \middle| D^{(s,m)}(\mathbf{x},t_{n-1})\right)} \tag{4.15}$$

where, the first term in the nominator is the conditional probability, or *likelihood*. It is a measure of our knowledge of the current frame observation that can be deduced from the previous frame observation and the parameter vector $\mathbf{p}$. The second term on the nominator is the *a priori* distribution of the parameters that reflect our knowledge about the actual parameters given the previous frame observation. The term in the denominator is the conditional probability of the current frame observation given the previous one.

Assuming that the track parameters and the frame observation at time $t_{n-1}$ are independent, we can write

$$p\left(\mathbf{p}^{(s,m)} \middle| D^{(s,m)}(\mathbf{x},t_{n-1})\right) = p\left(\mathbf{p}^{(s,m)}\right). \tag{4.16}$$

Since the probability in the denominator is not a function of the track parameters, it can be ignored and the MAP estimate of $\mathbf{p}^{(s,m)}$ will become

$$\hat{\mathbf{p}}^{(s,m)}(\mathbf{x}) = \arg\max_{\mathbf{p}} \left\{ p\left(D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(s,m)}, D^{(s,m)}(\mathbf{x},t_{n-1})\right) p\left(\mathbf{p}^{(s,m)}\right) \right\}. \tag{4.17}$$

Here, the first factor on the right is the *likelihood*. For subband $s$ of level $m$, the likelihood function of subpel $\mathbf{x}$ is the pdf of the differential noise signal $e^{(s,m)}(\mathbf{x})$ [59]:

$$p\left(D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(s,m)}, D^{(s,m)}(\mathbf{x},t_{n-1})\right) = p\left(e^{(s,m)}(\mathbf{x})\right). \tag{4.18}$$

Using Equation (3.31) and substituting for the CDWT subband differential noise pdf we obtain

$$p\left(D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(s,m)}, D^{(s,m)}(\mathbf{x},t_{n-1})\right) \propto \exp\left\{ -\frac{\left|e^{(s,m)}\right|^2}{2\sigma_{s,m}^2} \right\} \tag{4.19}$$

where $\sigma_{s,m}^2$ is the variance of the subband differential noise and is defined in Equation (3.32) (See Section 3.3.3).

The *maximum likelihood* (ML) estimate of the track parameters can then be obtained by substituting Equation (4.12) into Equation (4.19) and using the relation in Equation (3.32):

$$\hat{\mathbf{p}}^{(s,m)}(\mathbf{x}) = \arg\min_{\mathbf{p}} \left\{ \frac{SD^{(s,m)}(\mathbf{x},\mathbf{p}^{(s,m)})}{P^{(s,m)}} \right\} \tag{4.20}$$

where

$$SD^{(s,m)}(\mathbf{x},\mathbf{p}) = \left| D^{(s,m)}(\mathbf{x},t_{n-1}) - D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}),t_n) \right|^2 \tag{4.21}$$

is the "subband squared difference" (SSD) as defined by Magarey in [60] and $P^{(s,m)}$ denotes the *energy* of the corresponding wavelet filter $\psi^{(s,m)}$ of subband $s$ of level $m$ (See Section 3.3.2). There is a small difference between this definition and its original one. We changed the simple displacement parameter in the function arguments with the track parameter vector $\mathbf{p}$ and used the warping function $\mathbf{M}(\mathbf{x};\mathbf{p})$ instead of pure translation in the CDWT subband at time $t_n$.

#### 4.6.2.2 Parameter Estimate over Six Subbands

In order to extend this formulation over the six subbands, the joint CDWT noise pdf $p(\mathbf{e}^{(m)})$ is required [59], where $\mathbf{e}^{(m)} = [e^{(1,m)} \cdot \cdot \cdot \ e^{(6,m)}]^T$. Since the six subband filters of the CDWT are approximately disjoint in the frequency domain (See Figure 3.4), the noise in each subband is approximately independent of the noise in other subbands [59]. The joint pdf is therefore the product of the individual CDWT subband noise pdfs given in Equation (3.31):

$$p\left(\mathbf{e}^{(m)}\right) \propto \prod_{s=1}^{6} \exp\left\{ -\frac{\left| e^{(s,m)} \right|^2}{2\sigma_{s,m}^2} \right\} \tag{4.22}$$

Extending the likelihood given in Equation (4.18) to cover all the six subbands, we obtain

$$p\left( D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(m)}, D^{(s,m)}(\mathbf{x},t_{n-1}), s = 1,\ldots,6 \right) = p\left(\mathbf{e}^{(m)}(\mathbf{x})\right). \tag{4.23}$$

So, the likelihood will now be equal to the joint CDWT noise. Substituting (4.22) we obtain

$$p\left( D^{(s,m)}(\mathbf{x},t_n) \middle| \mathbf{p}^{(m)}, D^{(s,m)}(\mathbf{x},t_{n-1}), s = 1,\ldots,6 \right) \propto \prod_{s=1}^{6} \exp\left\{ -\frac{\left| e^{(s,m)} \right|^2}{2\sigma_{s,m}^2} \right\}. \tag{4.24}$$

Further propagating the product to the exponent as summation we can write

$$p\left(D^{(s,m)}(\mathbf{x},t_n)\middle|\mathbf{p}^{(m)},D^{(s,m)}(\mathbf{x},t_{n-1}),s=1,\ldots,6\right)\propto \exp\left\{-\frac{1}{2}\sum_{s=1}^{6}\frac{\left|e^{(s,m)}\right|^2}{\sigma_{s,m}^2}\right\}. \tag{4.25}$$

Then, the ML estimate over the six subbands will be

$$\hat{\mathbf{p}}^{(m)}(\mathbf{x})=\arg\min_{\mathbf{p}}\left\{SD^{(m)}(\mathbf{x},\mathbf{p}^{(m)})\right\} \tag{4.26}$$

where

$$SD^{(m)}(\mathbf{x},\mathbf{p})=\sum_{s=1}^{6}\frac{SD^{(s,m)}(\mathbf{x},\mathbf{p})}{P^{(s,m)}} \tag{4.27}$$

is the *combined* SSD surface [59]. Magarey approximated this surface by a second degree polynomial function using the CDWT subband coefficients of the two frames. He showed that the minimum of this surface corresponds to the displacement of the subpel. In fact, only two subbands (orientations) are enough to obtain a unique solution; however, all six are used to gain robustness [59].

#### 4.6.2.3 Parameter Estimate over a Region

Up to this point, we have a similar formulation with Magarey [59][60]. Since, we are interested in the set of pixels located within the track gate; we have to extend this to obtain a unique estimate over all target pixels (subpels). We assume that the parametric motion model fully describes the motion of the target pixels within the track gate. Under this assumption, the likelihood of $\mathbf{p}$ over the region $\mathbf{x}\in W^{(m)}(t_{n-1})$ can be written as

$$\begin{aligned} &p\left(D^{(s,m)}(\mathbf{x},t_n)\middle|\mathbf{p}^{(m)},D^{(s,m)}(\mathbf{x},t_{n-1}),s=1,\ldots,6,\mathbf{x}\in W^{(m)}(t_{n-1})\right)\\ &\qquad\qquad=p\left(e^{(m)}(\mathbf{x}),\mathbf{x}\in W^{(m)}(t_{n-1})\right). \end{aligned} \tag{4.28}$$

Assuming that noise is independent among neighboring pixels we can extend the noise distribution given in Equation (4.22) as

$$p\left(\mathbf{e}^{(m)}(\mathbf{x}),\mathbf{x}\in W^{(m)}(t)\right)\propto\prod_{\mathbf{x}\in W^{(m)}(t)}\prod_{s=1}^{6}\exp\left\{-\frac{\left|e^{(s,m)}\right|^2}{2\sigma_{s,m}^2}\right\} \tag{4.29}$$

Then, combining Equations (4.28) and (4.29), and converting the products to summations within the exponent, the ML estimate of **p** will be

$$\hat{\mathbf{p}}^{(m)}\left(W^{(m)}(t_{n-1})\right) = \arg\min_{\mathbf{p}}\left\{ \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} w^{(m)}(\mathbf{x})SD^{(m)}(\mathbf{x},\mathbf{p}^{(m)}) \right\}. \tag{4.30}$$

Here, we included also a weighting function $w^{(m)}(\mathbf{x})$ in order to be able to weight the subpels within the target region according to some criterion. The use of the weighting function will be explained later in Section 4.7.5. For simplicity, it will be assumed to be equal to 1.

## 4.6.3 The Objective Function

The tracking problem therefore converges to finding the minimizing argument $\mathbf{p}^{(m)}$ of the objective function

$$O^{(m)}\left(\mathbf{p}^{(m)}\right) = \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} w^{(m)}(\mathbf{x})SD^{(m)}(\mathbf{x},\mathbf{p}^{(m)}). \tag{4.31}$$

Substituting the definitions (4.27) and (4.21) into the objective function we obtain

$$O^{(m)}\left(\mathbf{p}^{(m)}\right) = \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} w^{(m)}(\mathbf{x}) \sum_{s=1}^{6} \frac{SD^{(s,m)}(\mathbf{x},\mathbf{p}^{(m)})}{P^{(s,m)}} \tag{4.32}$$

$$O^{(m)}\left(\mathbf{p}^{(m)}\right) = \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} w^{(m)}(\mathbf{x}) \sum_{s=1}^{6} \frac{\left|D^{(s,m)}(\mathbf{x},t_{n-1}) - D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n)\right|^2}{P^{(s,m)}}. \tag{4.33}$$

Rearranging the terms will yield

$$O^{(m)}\left(\mathbf{p}^{(m)}\right) = \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left|D^{(s,m)}(\mathbf{x},t_{n-1}) - D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n)\right|^2. \tag{4.34}$$

This objective function is also similar to the *subband squared difference* (SSD) definition of Magarey in his CDWT based motion estimation formulation in [59], except by the inclusion of an extra summation term and the use of the warping function **M** instead of a displacement parameter. This allows us to proceed in a similar way as in [59].

### 4.6.4 Minimization of the Objective Function

Expanding the objective function (Equation (4.34)) further we obtain

$$\begin{aligned} O^{(m)}(\mathbf{p}^{(m)}) = & \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left| D^{(s,m)}(\mathbf{x},t_{n-1}) \right|^2 + \left| D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n) \right|^2 \right] \\ & -2\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \mathrm{Re}\left\{ D^{*(s,m)}(\mathbf{x},t_{n-1}) D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n) \right\} \end{aligned} \tag{4.35}$$

Referring to the *shiftability* property of the CDWT coefficients explained in Section 3.3.1 we can rewrite the *staircase interpolation formula* given in Equations (3.27) as

$$D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p})) \approx D^{(s,m)}(\mathbf{x}) e^{j\theta^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p})-\mathbf{x})} \tag{4.36}$$

or, using the differential warp defined in Equation (4.8), as

$$D^{(s,m)}(\mathbf{x} + \tilde{\mathbf{M}}(\mathbf{x};\mathbf{p})) \approx D^{(s,m)}(\mathbf{x}) e^{j\theta^{(s,m)}(\tilde{\mathbf{M}}(\mathbf{x};\mathbf{p}))}\,. \tag{4.37}$$

Assuming that $\left| D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n) \right|^2$ in Equation (4.35) stays relatively constant with $\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)})$, then, minimizing the objective function $O^{(m)}(\mathbf{p}^{(m)})$ implies maximizing the cross-correlation term [59]. Using the staircase interpolation formula (4.36) in Equation (4.35) will yield

$$\begin{aligned} O^{(m)}(\mathbf{p}^{(m)}) \approx & \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left| D^{(s,m)}(\mathbf{x},t_{n-1}) \right|^2 + \left| D^{(s,m)}(\mathbf{x},t_n) \right|^2 \right] \\ & -2\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \mathrm{Re}\left\{ D^{*(s,m)}(\mathbf{x},t_{n-1}) D^{(s,m)}(\mathbf{x},t_n) e^{j\theta^{(s,m)}(\tilde{\mathbf{M}}(\mathbf{x};\mathbf{p}^{(m)}))} \right\} \end{aligned} \tag{4.38}$$

Evaluating the second term by taking the real part of complex exponential we obtain

$$\begin{aligned} O^{(m)}(\mathbf{p}^{(m)}) \approx & \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left| D^{(s,m)}(\mathbf{x},t_{n-1}) \right|^2 + \left| D^{(s,m)}(\mathbf{x},t_n) \right|^2 \right] \\ & -2\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left| D^{(s,m)}(\mathbf{x},t_{n-1}) D^{(s,m)}(\mathbf{x},t_n) \right| \cos\left( \theta^{(s,m)}(\tilde{\mathbf{M}}(\mathbf{x};\mathbf{p}^{(m)})) \right) \end{aligned} \tag{4.39}$$

Expanding the phase term will yield

$$
\begin{aligned}
O^{(m)}(\mathbf{p}^{(m)}) \approx \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left|D^{(s,m)}(\mathbf{x},t_{n-1})\right|^2 + \left|D^{(s,m)}(\mathbf{x},t_n)\right|^2 \right] \\
-2\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left|D^{(s,m)}(\mathbf{x},t_{n-1})D^{(s,m)}(\mathbf{x},t_n)\right| \\
\cos\left(\phi^{(s,m)}(\mathbf{x},t_{n-1}) - \phi^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n)\right)
\end{aligned}
\tag{4.40}
$$

for $\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}^{(m)}) \in [-0.5,0.5]\times[-0.5,0.5]$ and for all $\mathbf{x}\in W^{(m)}(t_{n-1})$ where $\phi^{(s,m)}(\mathbf{x},t_{n-1})$ and $\phi^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n)$ are phases of $D^{(s,m)}(\mathbf{x},t_{n-1})$ and $D^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n)$, respectively. Then, the objective function can be minimized by maximizing the *phase correlation* between the two sets of subpels within the target region. The phase correlation of each subpel is weighted by the term called "activity" [59] which is defined as

$$
E^{(s,m)}(\mathbf{x};t_{n-1},t_n) = \left|D^{(s,m)}(\mathbf{x},t_{n-1})D^{(s,m)}(\mathbf{x},t_n)\right|. \tag{4.41}
$$

The minimum of the objective function is specified by the *equiphase equation* [59]:

$$
\phi^{(s,m)}(\mathbf{x},t_{n-1}) = \phi^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(m)}),t_n) \tag{4.42}
$$

which should be solved for $\mathbf{p}^{(m)}$ over $s$ = 1,…,6 and over all $\mathbf{x}\in W^{(m)}(t_{n-1})$. This equiphase equation is similar to the one in the solution of CDWT-ME [59], with only a difference in the displacement term. Using the planar model in Equations (4.36) and (3.28) we can write

$$
2^m \left(\Omega^{(s,m)}\right)^T \widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}) = \theta^{(s,m)}(\mathbf{x}). \tag{4.43}
$$

where

$$
\theta^{(s,m)}(\mathbf{x}) = \angle\left[\frac{D^{(s,m)}(\mathbf{x},t_n)}{D^{(s,m)}(\mathbf{x},t_{n-1})}\right] \tag{4.44}
$$

and $\Omega^{(s,m)}$ is the *center frequency* of the corresponding wavelet filter (See Equation (3.15)). Since the CDWT pyramid is perfectly scaled the components of $2^m\Omega^{(s,m)}$ are independent of $m$ [59]:

$$
2^m \left(\Omega^{(m,s)}\right)^T = \left[\Omega_1^{(s)} \quad \Omega_2^{(s)}\right] \tag{4.45}
$$

Substituting (4.43) into Equation (4.39) and further approximating the objective function using $\cos(x) \approx 1 - \frac{x^2}{2}$ we obtain [59]

$$O^{(m)}(\mathbf{p}^{(m)}) \approx \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left| D^{(s,m)}(\mathbf{x}, t_{n-1}) \right| - \left| D^{(s,m)}(\mathbf{x}, t_n) \right| \right]^2 + \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left| D^{(s,m)}(\mathbf{x}, t_{n-1}) D^{(s,m)}(\mathbf{x}, t_n) \right| \left[ 2^m \left( \boldsymbol{\Omega}^{(m,s)} \right)^T \tilde{\mathbf{M}}(\mathbf{x}; \mathbf{p}^{(m)}) - \theta^{(s,m)}(\mathbf{x}) \right]^2 \tag{4.46}$$

This equation is again similar to Magarey's solution [59]. We have an extra summation term and have used the warping function instead of a displacement term. The objective function can be characterized by a second order *quadratic* surface [59]. The solution of this objective function can be found by differentiating the objective function with respect to each parameter and equating them to zero:

$$\nabla O^{(m)}(\mathbf{p}^{(m)}) = 0 \tag{4.47}$$

where

$$\nabla O^{(m)}(\mathbf{p}^{(m)}) = \begin{bmatrix} \frac{\partial}{\partial p_1} O^{(m)}(\mathbf{p}^{(m)}) \\ \vdots \\ \frac{\partial}{\partial p_k} O^{(m)}(\mathbf{p}^{(m)}) \end{bmatrix}. \tag{4.48}$$

If we define

$$\tilde{\mathbf{M}}(\mathbf{x}; \mathbf{p}) = \begin{bmatrix} \tilde{M}_1(\mathbf{x}; \mathbf{p}) \\ \tilde{M}_2(\mathbf{x}; \mathbf{p}) \end{bmatrix} \tag{4.49}$$

and use (4.45) we may write Equation (4.46) as

$$O^{(m)}(\mathbf{p}^{(m)}) \approx \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left[ \left| D^{(s,m)}(\mathbf{x}, t_{n-1}) \right| - \left| D^{(s,m)}(\mathbf{x}, t_n) \right| \right]^2 + \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} \left| D^{(s,m)}(\mathbf{x}, t_{n-1}) D^{(s,m)}(\mathbf{x}, t_n) \right| \left[ \Omega_1 \tilde{M}_1(\mathbf{x}; \mathbf{p}^{(m)}) + \Omega_2 \tilde{M}_2(\mathbf{x}; \mathbf{p}^{(m)}) - \theta^{(s,m)}(\mathbf{x}) \right]^2 \tag{4.50}$$

Then, the solution (4.47) could be obtained using the chain rule

$$\nabla O^{(m)}(\mathbf{p}^{(m)}) = \left[\frac{\partial}{\partial \widetilde{M}_1} O^{(m)}(\mathbf{p}^{(m)})\right] \nabla \widetilde{M}_1(\mathbf{x};\mathbf{p}^{(m)}) + \left[\frac{\partial}{\partial \widetilde{M}_2} O^{(m)}(\mathbf{p}^{(m)})\right] \nabla \widetilde{M}_2(\mathbf{x};\mathbf{p}^{(m)}) . \quad (4.51)$$

Substituting Equation (4.50) into (4.51) and ignoring the constant 2 coming from the derivatives will yield

$$\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x},t_{n-1},t_n)\left[\Omega_1 \widetilde{M}_1(\mathbf{x};\mathbf{p}^{(m)}) + \Omega_2 \widetilde{M}_2(\mathbf{x};\mathbf{p}^{(m)}) - \theta^{(s,m)}(\mathbf{x})\right] \left(\Omega_1 \nabla \widetilde{M}_1(\mathbf{x};\mathbf{p}^{(m)}) + \Omega_2 \nabla \widetilde{M}_2(\mathbf{x};\mathbf{p}^{(m)})\right) = 0 \quad (4.52)$$

whose solution depends on the motion model **M**(**x**;**p**). Solutions for three types of models are given in the next Section.

### 4.6.5 Solutions According to Motion Models

Closer inspection of Equation (4.52) shows that it can be converted to a set of linear equations in the form:

$$\mathbf{Tp} = \mathbf{a} \quad (4.53)$$

whose solution can be obtained by inverting the coefficient matrix **T**:

$$\mathbf{p} = \mathbf{T}^{-1}\mathbf{a} \quad (4.54)$$

We introduced three types of motion models in Section 4.4 whose solutions are given below.

#### 4.6.5.1 Solution of the Translational Motion Model

In the *translational* motion model we have two parameters:

$$\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}) = \mathbf{M}(\mathbf{x};\mathbf{p}) - \mathbf{x} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}. \quad (4.55)$$

Substituting this model into Equation (4.52) we obtain

$$\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x},t_{n-1},t_n)\left[\Omega_1 p_1 + \Omega_2 p_2 - \theta^{(s,m)}(\mathbf{x})\right] \begin{bmatrix} \Omega_1 \\ \Omega_2 \end{bmatrix} = 0 . \quad (4.56)$$

By rearranging the terms as in the form given in (4.53) we obtain

$$\mathbf{T} = \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \begin{bmatrix} \Omega_1^2 & \Omega_1\Omega_2 \\ \Omega_1\Omega_2 & \Omega_2^2 \end{bmatrix} \tag{4.57}$$

$$\mathbf{p} = [p_1 \quad p_2]^T \tag{4.58}$$

$$\mathbf{a} = \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \theta^{(s,m)}(\mathbf{x}) \begin{bmatrix} \Omega_1 \\ \Omega_2 \end{bmatrix}. \tag{4.59}$$

The solution of these equations for (4.58) gives the estimate of the track parameters **p**. Note that, if we limit the target region to one pixel, then we obtain the same solution for a single pixel as Magarey [59][60].

#### 4.6.5.2 Solution of the Similarity Motion Model

For the *similarity* motion model we have four parameters:

$$\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}) = \mathbf{M}(\mathbf{x};\mathbf{p}) - \mathbf{x} = \begin{bmatrix} p_1 x_1 - p_2 x_2 + p_3 \\ p_2 x_1 + p_1 x_2 + p_4 \end{bmatrix} \tag{4.60}$$

Substituting this into Equation (4.52) we obtain

$$\sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n)$$
$$\left[\Omega_1(p_1x_1 - p_2x_2 + p_3) + \Omega_2(p_2x_1 + p_1x_2 + p_4) - \theta^{(s,m)}(\mathbf{x})\right] \begin{bmatrix} \Omega_1x_1 + \Omega_2x_2 \\ \Omega_2x_1 - \Omega_1x_2 \\ \Omega_1 \\ \Omega_2 \end{bmatrix} = 0 \tag{4.61}$$

Then, using the form in Equation (4.53)

$$\mathbf{T} = \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \begin{bmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{V}^T & \mathbf{Z} \end{bmatrix} \tag{4.62}$$

$$\mathbf{p} = [p_1 \quad p_2 \quad p_3 \quad p_4]^T \tag{4.63}$$

$$\mathbf{a} = \sum_{s=1}^{6} \sum_{\mathbf{x}\in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \theta^{(s,m)}(\mathbf{x}) \begin{bmatrix} \Omega_1x_1 + \Omega_2x_2 \\ \Omega_2x_1 - \Omega_1x_2 \\ \Omega_1 \\ \Omega_2 \end{bmatrix} \tag{4.64}$$

where

$$\mathbf{U} = \begin{bmatrix} (\Omega_1 x_1 + \Omega_2 x_2)^2 & (\Omega_1 x_1 + \Omega_2 x_2)(\Omega_2 x_1 - \Omega_1 x_2) \\ (\Omega_1 x_1 + \Omega_2 x_2)(\Omega_2 x_1 - \Omega_1 x_2) & (\Omega_2 x_1 - \Omega_1 x_2)^2 \end{bmatrix} \tag{4.65}$$

$$\mathbf{V} = \begin{bmatrix} \Omega_1(\Omega_1 x_1 + \Omega_2 x_2) & \Omega_2(\Omega_1 x_1 + \Omega_2 x_2) \\ \Omega_1(\Omega_2 x_1 - \Omega_1 x_2) & \Omega_2(\Omega_2 x_1 - \Omega_1 x_2) \end{bmatrix} \tag{4.66}$$

$$\mathbf{Z} = \begin{bmatrix} (\Omega_1)^2 & (\Omega_1 \Omega_2) \\ (\Omega_1 \Omega_2) & (\Omega_2)^2 \end{bmatrix} \tag{4.67}$$

The solution of these equations will give the track parameter estimates of the *similarity* motion model.

#### 4.6.5.3 Solution of the Affine Motion Model

For the *affine* motion model we have six parameters:

$$\widetilde{\mathbf{M}}(\mathbf{x};\mathbf{p}) = \mathbf{M}(\mathbf{x};\mathbf{p}) - \mathbf{x} = \begin{bmatrix} p_1 x_1 + p_3 x_2 + p_5 \\ p_2 x_1 + p_4 x_2 + p_6 \end{bmatrix} \tag{4.68}$$

Using this model in Equation (4.52) we obtain

$$\sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n)$$

$$\left[\Omega_1 (p_1 x_1 + p_3 x_2 + p_5) + \Omega_2 (p_2 x_1 + p_4 x_2 + p_6) - \theta^{(s,m)}(\mathbf{x})\right] \begin{bmatrix} \Omega_1 x_1 \\ \Omega_2 x_1 \\ \Omega_1 x_2 \\ \Omega_2 x_2 \\ \Omega_1 \\ \Omega_2 \end{bmatrix} = 0 \tag{4.69}$$

Arranging the terms as in the form given in Equation (4.53) we obtain the following equations:

$$\mathbf{T} = \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \begin{bmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{V}^T & \mathbf{Z} \end{bmatrix} \tag{4.70}$$

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{bmatrix}^T \tag{4.71}$$

$$\mathbf{a} = \sum_{s=1}^{6} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} \frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}} E^{(s,m)}(\mathbf{x}, t_{n-1}, t_n) \theta^{(s,m)}(\mathbf{x}) \begin{bmatrix} \Omega_1 x_1 \\ \Omega_2 x_1 \\ \Omega_1 x_2 \\ \Omega_2 x_2 \\ \Omega_1 \\ \Omega_2 \end{bmatrix} \tag{4.72}$$

where

$$\mathbf{U} = \begin{bmatrix} (\Omega_1 x_1)^2 & (\Omega_1 \Omega_2 x_1^2) & (\Omega_1^2 x_1 x_2) & (\Omega_1 \Omega_2 x_1 x_2) \\ (\Omega_1 \Omega_2 x_1^2) & (\Omega_2 x_1)^2 & (\Omega_1 \Omega_2 x_1 x_2) & (\Omega_2^2 x_1 x_2) \\ (\Omega_1^2 x_1 x_2) & (\Omega_1 \Omega_2 x_1 x_2) & (\Omega_1 x_2)^2 & (\Omega_1 \Omega_2 x_2^2) \\ (\Omega_1 \Omega_2 x_1 x_2) & (\Omega_2^2 x_1 x_2) & (\Omega_1 \Omega_2 x_2^2) & (\Omega_2 x_2)^2 \end{bmatrix} \tag{4.73}$$

$$\mathbf{V} = \begin{bmatrix} (\Omega_1^2 x_1) & (\Omega_1 \Omega_2 x_1) \\ (\Omega_1 \Omega_2 x_1) & (\Omega_2^2 x_1) \\ (\Omega_1^2 x_2) & (\Omega_1 \Omega_2 x_2) \\ (\Omega_1 \Omega_2 x_2) & (\Omega_2^2 x_2) \end{bmatrix} \tag{4.74}$$

$$\mathbf{Z} = \begin{bmatrix} (\Omega_1)^2 & (\Omega_1 \Omega_2) \\ (\Omega_1 \Omega_2) & (\Omega_2)^2 \end{bmatrix} \tag{4.75}$$

The solution of these equations for **p** will give the estimate of the track parameter vector **p** given in Equation (4.71).

## 4.7 Hierarchical Tracking

Parameter estimation starts at the coarsest level and is refined hierarchically in a similar manner as the motion estimation algorithm of Magarey [59][60]. The track parameters **T**, **p**, and **a** have to be passed to the next finer level in an appropriate way to constitute the *cumulative* track parameters. The steps are summarized in Section 4.5. This section describes each step in detail.

### 4.7.1 Scaling of the Track Parameters

The *cumulative* track parameters $\overline{\mathbf{T}}$, $\overline{\mathbf{p}}$, and $\bar{\mathbf{a}}$ need to be scaled from the previous level to the current level in order to accommodate for scale changes due to the change in resolution. The coordinate frame is scaled by $2^{-m}$. So, while moving at a finer level, the $x_1$ and $x_2$ should be scaled by 2.

### 4.7.1.1 Scaling of the Translational Motion Model Parameters

The track parameters **T**, **p**, and **a** for the *translational* motion model have to be scaled as follows:

$$\begin{aligned} \mathbf{T} &\rightarrow \mathbf{T} / 4 \\ \mathbf{p} &\rightarrow 2\mathbf{p} \\ \mathbf{a} &\rightarrow \mathbf{a} / 2 \end{aligned} \tag{4.76}$$

In the *translational* motion model, the parameters **p** are pure translation and depend upon the coordinates. Therefore they should be scaled by 2.

### 4.7.1.2 Scaling of the Similarity Motion Model Parameters

The track parameters **T**, **p**, and **a** for the *similarity* motion model have to be scaled as follows:

$$\begin{aligned} \mathbf{U} &\rightarrow \mathbf{U} \\ \mathbf{V} &\rightarrow \mathbf{V} / 2 \\ \mathbf{Z} &\rightarrow \mathbf{Z} / 4 \\ a_1, a_2 &\rightarrow a_1, a_2 \\ a_3, a_4 &\rightarrow a_3 / 2, a_4 / 2 \\ p_1, p_2 &\rightarrow p_1, p_2 \\ p_3, p_4 &\rightarrow 2p_3, 2p_4 \end{aligned} \tag{4.77}$$

Here, **U**, **V**, and **Z** are submatrices of **T** as defined in Equation (4.62), and, $a_i$ and $p_i$ are elements of the vectors **a** and **p**. In this model, $p_1$ and $p_2$ are elements of the deformation matrix therefore, they remain unchanged. However, as in the *translational* motion model, $p_3$ and $p_4$ refer to the displacement and should be scaled by 2.

#### 4.7.1.3 Scaling of the Affine Motion Model Parameters

The track parameters **T**, **p**, and **a** for the *affine* motion model have to be scaled as follows:

$$\begin{aligned}
\mathbf{U} &\rightarrow \mathbf{U} \\
\mathbf{V} &\rightarrow \mathbf{V}/2 \\
\mathbf{Z} &\rightarrow \mathbf{Z}/4 \\
a_1, a_2, a_3, a_4 &\rightarrow a_1, a_2, a_3, a_4 \\
a_5, a_6 &\rightarrow a_5/2, a_6/2 \\
p_1, p_2, p_3, p_4 &\rightarrow p_1, p_2, p_3, p_4 \\
p_5, p_6 &\rightarrow 2p_5, 2p_6
\end{aligned} \tag{4.78}$$

Here, **U**, **V**, and **Z** are submatrices of **T** as defined in Equation (4.70), and, $a_i$ and $p_i$ are elements of the vectors **a** and **p**. For the model, $p_1 - p_4$ are elements of the deformation matrix therefore, they remain unchanged. However, as in the *translational* motion model, $p_5$ and $p_6$ refer to the displacement and should be scaled by 2.

### 4.7.2 Warping of CDWT Subband Coefficients

The CDWT subbands of frame at time $t_{n-1}$ are pre-warped according to the cumulative track parameters **p** estimated at the previous levels and scaled to this level. This pre-warping has a great effect in the precision of the algorithm. It considerably simplifies the matching process at finer levels. The windowed sinc interpolator (Section 3.3.1.1) is used to obtain the CDWT coefficients at non-integer coordinates. This deformation is inverted back as explained in the next Section.

### 4.7.3 Recovering Original Track Parameters

After the track parameters **p** are estimated for this level, the effect of pre-warping the CDWT subband coefficients should be inverted back in order to obtain original **p** values rather than relative ones. This is achieved by adjusting the parameters **a** as follows:

$$\mathbf{a} \;\rightarrow\; \mathbf{a} + \mathbf{Tp} \tag{4.79}$$

where **T** is the one obtained for this current level, but **p** is the cumulative one used for pre-warping the CDWT subband as explained in the previous section. This modification eliminates the effect of the pre-warping process.

### 4.7.4 Accumulating the Track Parameters

The parameters computed at each level are added in order to form the cumulative parameters and obtain a unique, combined estimate of the track parameters **p**. After recovering original parameters as explained in the previous Section the parameters **T** and **a** are accumulated as follows:

$$\overline{\mathbf{T}}^{(m)} = \begin{cases} \overline{\mathbf{T}}^{(m+1)} + \mathbf{T}^{(m)} & m_{\max} \geq m > m_{\min} \\ \mathbf{T}^{(m)} & m = m_{\max} \end{cases} \tag{4.80}$$

$$\overline{\mathbf{a}}^{(m)} = \begin{cases} \overline{\mathbf{a}}^{(m+1)} + \mathbf{a}^{(m)} & m_{\max} \geq m > m_{\min} \\ \mathbf{a}^{(m)} & m = m_{\max} \end{cases} \tag{4.81}$$

Then, the solution to the equation **Tp** = **a** using the cumulative parameters $\overline{\mathbf{T}}$, $\overline{\mathbf{p}}$, and $\bar{\mathbf{a}}$ gives the new track parameters **p**.

### 4.7.5 Computing the Weights

The weights, $w^{(m)}(\mathbf{x})$, are used for several purposes. The first use is that they enable to select an arbitrary shape within the rectangular region, and hence, function as a *mask*. This provides ease of implementation as the CDWT is obtained for rectangular regions only. In this way, the weights are used to select only the complex wavelet coefficients of the target pixels that are within the affine track gate.

The second use is that they can provide coefficient weighting at the gate boundary while going to coarser resolution levels where one subpel corresponds to $2^m \times 2^m$ pixels, where possibly not all of them would be target pixels.

A third use could be a windowing function, or, a kernel in order to give more weights to the center of the track gate. A predefined kernel could be used for this purpose.

Another use would be to provide an arbitrary shape to describe the target. Since, the rectangular track gate rarely would fit completely to the target and, if available, a description of the shape of the target would be helpful to aid the tracker. So, the weights could be adjusted according to the shape.

## 4.8 Summary

In this Chapter, the Complex Wavelet Tracker is presented. Although developed for the *translational* motion model, the CDWT can handle affine deformations to some extend, on which we based our method. Different parametric motion models can be used to model the motion of the track gate. We define tracking as the estimation of these parameters throughout the tracking sequence. The method is hierarchical in structure and the parameters estimated in one level are propagated into the finer level. Solutions according to *translational*, *similarity* and *affine* motion models are provided at the end of this Chapter.

# CHAPTER 5

# EXTENSIONS AND OTHER APPLICATION AREAS

This Chapter captures some extensions to the base algorithm and explores other application areas that the proposed method could be used for. The first section presents some extension and modifications to the base method. Based on these modifications, other application areas other than tracking are presented. A complete multi-target tracking framework combining different aspects of tracking systems is introduced in the last section of this Chapter.

## 5.1 Extensions to the CWT

This section coveres several extension and modifications to the base algorithm. The purpose is to improve and extend the application areas of the Complex Wavelet Tracker.

### 5.1.1 Color Tracking

The Complex Wavelet Tracker operates on single intensity values. For color video, it is sufficient for most cases to convert the color information to grayscale and perform the tracking on intensities. For some other cases, however, color information could be more distinguishing than simple intensity.

We can start in parallel with the derivation Magarey had performed for the CDWT based motion estimation algorithm [59] for color images. Let the input image, $A(\mathbf{x},t)$ be a vector image $\mathbf{A}(\mathbf{x},t)$ with three color components:

$$\mathbf{A}(\mathbf{x},t) = \begin{bmatrix} A_1(\mathbf{x},t) \\ A_2(\mathbf{x},t) \\ A_3(\mathbf{x},t) \end{bmatrix} \tag{5.1}$$

The CDWT of all components should be computed separately. The detailed subimages would then be also in vector form:

$$\mathbf{D}^{(s,m)}(\mathbf{x},t) = \begin{bmatrix} D_1^{(s,m)}(\mathbf{x},t) \\ D_2^{(s,m)}(\mathbf{x},t) \\ D_3^{(s,m)}(\mathbf{x},t) \end{bmatrix} \tag{5.2}$$

In the same way, the differential subband noise in (4.12) will also be in vector form as follows:

$$\mathbf{e}^{(s,m)}(\mathbf{x}) = \begin{bmatrix} D_1^{(s,m)}(\mathbf{x},t_{n-1}) - D_1^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(s,m)}),t_n) \\ D_2^{(s,m)}(\mathbf{x},t_{n-1}) - D_2^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(s,m)}),t_n) \\ D_3^{(s,m)}(\mathbf{x},t_{n-1}) - D_3^{(s,m)}(\mathbf{M}(\mathbf{x};\mathbf{p}^{(s,m)}),t_n) \end{bmatrix} \tag{5.3}$$

Assuming white Gaussian noise in the vector images, the joint pdf of the differential subband noise can be written as

$$p\left(\mathbf{e}^{(s,m)}(\mathbf{x})\right) \propto \exp\left\{-\frac{1}{2}\left[\mathbf{e}^{(s,m)}\right]^H \left[\mathbf{R}^{(s,m)}\right]^{-1} \mathbf{e}^{(s,m)}\right\} \tag{5.4}$$

where $\mathbf{R}^{(s,m)}$ is the $3 \times 3$ component noise covariance matrix in subband $s$ of level $m$, and will be given by [59]

$$\mathbf{R}^{(s,m)} = P^{(s,m)}\,\mathbf{R} \tag{5.5}$$

where $P^{(s,m)}$ is the energy of the corresponding wavelet filter (See Section 3.3.2). Equation (5.5) states that inter-field noise statistics are unchanged by the CDWT, apart from a scaling by the corresponding subband energy [59].

Assuming that noise in each subband is approximately uncorrelated with that in other subbands, **R** will be diagonal [59]. Assuming further that the noise in each color component is uncorrelated then **R** will be [59]:

$$\mathbf{R} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix} \tag{5.6}$$

Continuing with the derivation, the objective function given in (4.31) will now be

$$O^{(m)}\left(\mathbf{p}^{(m)}\right) = \sum_{k=1}^{3} \sum_{\mathbf{x} \in W^{(m)}(t_{n-1})} w^{(m)}(\mathbf{x}) \frac{SD_k^{(m)}(\mathbf{x},\mathbf{p}^{(m)})}{\sigma_k^2} \tag{5.7}$$

where $SD_k^{(m)}$ is the SSD of the kth color component. To summarize, if the component noise is uncorrelated, then, an estimate can be obtained simply by the addition of the three color components' SSD surfaces, weighted by the inverse of the noise variance in that component.

## 5.1.2 Confidence Measure

With an analogy to the base work of Magarey [59], a confidence measure could be established in order to obtain a measure for the quality of the estimate.

The measure is based on the steepness of the quadratic surface. The closer the minimum lines of the six subbands, the steeper the surface will be. When the minimum lines of each subband are far apart, as will be in the case when the signals are unrelated, then the surface will be spread [59]. The measure will therefore be a measure of the spread of the minimum lines and will be used to define the confidence measure.

This will not change when applied to regions. Since, according to the derivations, the solution is obtained from the same surface. So, the steepness of this surface could reflect the confidence for the proposed method as well.

The *confidence* measure is defined as [59]

$$C^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n) = 1 - \frac{r^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n)}{2} \tag{5.8}$$

where $W^{(m)}$ is the set of subpels in level $m$. The *residual* $r^{(m)}$ is defines as [59]

$$r^{(m)}(W^{(m)}) = \frac{\delta - \Delta^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n)}{E^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n)} \tag{5.9}$$

and is normalized by the activity

$$E^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n)=\sum_{s=1}^{6}\sum_{\mathbf{x}\in W^{(m)}(t_{n-1})}\frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}}E^{(s,m)}(\mathbf{x};t_{n-1},t_n) \tag{5.10}$$

where $w^{(m)}(\mathbf{x})$ are the weights and $E^{(s,m)}(\mathbf{x};t_{n-1},t_n)$ is defined in (4.41). The *discrepancy* [59], $\Delta$, in (5.9) is the first term on the right of Equation (4.50) and is defined as

$$\Delta^{(m)}(W^{(m)}(t_{n-1});t_{n-1},t_n)=\sum_{s=1}^{6}\sum_{\mathbf{x}\in W^{(m)}(t_{n-1})}\frac{w^{(m)}(\mathbf{x})}{P^{(s,m)}}\left[\left|D^{(s,m)}(\mathbf{x},t_{n-1})\right|-\left|D^{(s,m)}(\mathbf{x},t_n)\right|\right]^2 \tag{5.11}$$

The term $\delta$ in (5.9) is the *surface minimum* value and can be obtained from the objective function as follows [59]:

$$\delta=O^{(m)}\left(\mathbf{p}_0^{(m)}\right) \tag{5.12}$$

where $\mathbf{p}_0^{(m)}$ is the track parameter vector that minimized the objective function. The range of the confidence measure is $(-\infty, 1]$.

### 5.1.3 Tracking by using a Target Template

The track gate is continuously updated throughout the tracking sequence and the new pixels within the estimated track gate are used as a basis for the next estimation. Assuming that the set of pixels within the current track gate is the target template, then, this template is updated at every frame as it is completely replaced with the previous one. This is desired for the purpose of this thesis where the target is allowed to undergo drastic changes in its appearance, and any limitation to this, might result in a degradation of the algorithm. However, for other applications, where the target signature remains stable within some limits, then, a fixed, or slowly updating template would be more preferable.

For this purpose, two approaches are possible. For the first one, the CDWT of the target within the track gate on the first frame is used as the template. For every estimation, instead of using the previous frame observation, the coefficients of the template are used. The track gate is updated in a same way according to the estimated track parameters. This is the *non-warping* approach. Here, the template is held fixed. To increase performance, the location of the template could be updated using the translational component of the estimated track parameters, if necessary.

The second approach is the *warping* approach. In this case, the template itself is also warped according to the estimated track parameters. This time, the CDWT coefficients could not be used directly and should be computed from the warped template for every frame.

The first approach should be used when the total deformation is limited to the affine deformation limits that the CDWT can handle. These limits are explored in Section 0. Otherwise, the *warping* approach should be used to handle more extensive deformations.

For the second approach, instead of keeping the template image totally fixed, the template could be smoothed in time using previous and new target templates. The filtering should be applied on the warpped template images in order to have a proper spatial match.

### 5.1.4 Tracking on a Fixed Track Gate

The track gate is continuously updated throughout the tracking sequence and the new pixels within the estimated track gate are used as a basis for the next estimation. Instead of updating the track gate, the parameters could be passed to another application and the position and shape of the track gate could remain fixed.

This can be viewed as a window (the track gate) behind which information is flowing. Since the window is fixed, in this way, we obtain the motion information of what is moving through the window.

This type of operation could be required for some motion estimation applications such as global motion estimation or scene stabilization where small windows throughout the image are used to obtain motion information and in this way extract the cumulative (global) motion of the image itself.

### 5.1.5 Fusion with Intensity-Based Methods

Fusion with an appearance- or intensity-based method would improve performance. The image registration method of Lucas and Kanade [57] using the same motion model could be used for this purpose.

The robustness of the Complex Wavelet Tracker and the accuracy of the Lucas-Kanade could be merged to obtain a superior algorithm that combines spatial and frequency domain methods. This can be established in the following way:

The fusion is based on the hierarchical structure of the CDWT. As the CDWT is pyramidal, the hierarchical version of Lucas-Kanade [57] should be employed. For every resolution level, Lucas-Kanade algorithm is performed on the lowpass subimage, $I^{(1,m)}$, of the CDWT pyramid. These estimates can be combined with the estimates of the Complex Wavelet Tracker in a number of ways. Either the estimates can be merged at every level, or, merging can be performed at the highest level only. Different merging strategies can be applied to both cases.

## 5.2 Other Application Areas

This section will cover application areas other than tracking for the proposed algorithm. The application may use the Complex Wavelet Tracker directly or benefit from the extensions mentioned in the previous section.

### 5.2.1 Generalized Block Motion Estimation

The Complex Wavelet Tracker is a base algorithm developed for area tracking. Hence, simulations and tests are directed to this purpose. However, this base also directly contributes to the area of generalized or deformable block motion estimation. Generalized block motion estimation is the generalization of the translational block motion estimation (See Section 2.1.2).

The most popular use of block motion estimation is for digital video compression (H.261 and MPEG-1, MPEG-2). The current frame is divided into blocks where each block is to be constructed from other frames. Both uniform and non-uniform grids could be used to tile the frame into blocks. Also, different shapes other than rectangular blocks can be used for the block to best suit the application. Blocks can be triangular, rectangular, hexagonal, or arbitrary quadrilateral in shape.

Block motion estimation is then performed on each block using previous or feature frames. For coding, the motion parameters for each block are used instead of the intensity information of the block itself.

The Complex Wavelet Tracker could be used to estimate the motion parameters for each block. For this purpose, the track gate should be initiated at each block. The weights can be used as a mask to match the arbitrary shape of the block. Since, the complexity of the Complex Wavelet Tracker is $O(N)$, the size and number of the blocks will have little effect on the overall computation time.

### 5.2.2 Global Motion Estimation and Background Stabilization

Global Motion Estimation (GME) refers to the estimation of the relative motion among successive frames. This motion is generally caused by the imaging source itself due to motions in the imaging source or the platform itself. In this way, it is also referred to as *ego-motion*.

One purpose to perform GME is to understand the motion caused by the camera or the platform itself. And the other would be to stabilize the image which is referred to as background stabilization. For either purpose, the estimation effort is the same.

GME can be performed in two ways using the Complex Wavelet Transform by using the fixed track gate solution explained in Section 5.1.4. The first way would be to use the whole frame as the track gate and perform the estimation. This approach would be more suitable for backgrounds where there would be no moving objects present disturbing the overall motion of the scene.

A solution would be to use a mask to eliminate moving objects interfering the solution. For this case, however, the detection of the moving objects is required. It would be sufficient if a subregion could be determined where the moving objects will be present. Then, this subregion could be masked out from the solution using the weights.

However, using the whole frame to obtain a solution could be costly in complexity. To overcome this, the image could be first downsized to a more appropriate size and estimation could be performed on this resized image. This would also eliminate the effect of small foreground moving objects as their sizes would also be downsized. Since the proposed method has sub-pixel accuracy this resizing would have a negligible effect on the estimation, as far as enough patterns remain in the downsized image for confident estimation.

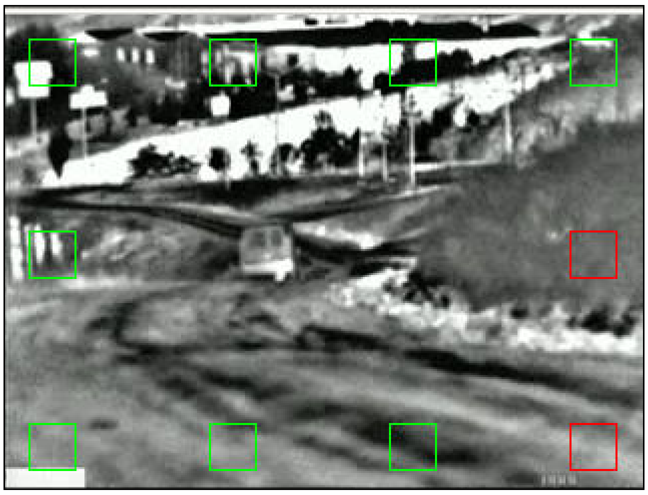


**Figure 5.1** Background stabilization can be performed using small track gates and by computing the overall motion from the individual displacement estimates of the gates.

The other way for GME would be to use small gates near the border of the frame as shown in Figure 5.1. Then, the overall motion could be computed from the individual displacement estimates of the gates. The translational component of each gate would be enough for the solution. A parametric motion model can then be employed using the gates. The solution of this model can be obtained using least squares estimation. So, in order to obtain a unique solution, at least one gate is required for the translational motion model, and a minimum of three gates for the affine motion model. Note that, more gates than the minimum required would increase the robustness of the solution.

The confidence measure described in Section 5.1.2 could be used eliminate non-confident gates to effect the overall solution. These gate are shown with a red gate in Figure 5.1.

### 5.2.3 Feature Tracking

The Complex Wavelet Tracker can easily be used to track point-features like the Shi-Tomasi's feature tracker[5] in [79]. Features can be selected in the same way and a track gate be opened on each feature. The size of the features can be 8 × 8 pixels, for which a 2 or 3 level pyramid would be enough. Both affine and *translational* motion models could be used to track and/or monitor track quality.

For tracking the features, the Complex Wavelet Tracker could either be used directly, or, the target template extension explained in Section 5.1.3 could be used, depending on the application. The latter would be more preferable for applications similar to [79] where point features over the whole frame are extracted and tracked to obtain information for the motion of the camera itself.

### 5.2.4 Face, Eye, and Car Tracking

Face tracking, eye tracking, and car tracking are special areas for tracking. The shapes are known beforehand, or are set at the beginning of the tracking. For these kinds of applications, using a fixed template would improve the performance. Hence, the Complex Wavelet Tracker with the target template extension explained in Section 5.1.3 could be used for such applications.

### 5.2.5 Image Alignment, Image Mosaicking, and Image Registration

Image alignment, image mosaicking, and image registration techniques require the matching of image regions. The Complex Wavelet Tracker could be used directly by opening setting the track gate on the region of interest on the first image and estimating the warp parameters for the second.

If the non-overlapping regions of the frames are larger than the Complex Wavelet Tracker can handle, then a two-stage algorithm could be used. In the first stage, a

[5] This feature tracker is also known as the "Kanade-Lucas-Tomasi (KLT) Feature Tracker".

rough alignment of the two images could be performed using some other simpler methods. In the second stage, the Complex Wavelet Tracker could be employed to fine adjust the warping parameters.

## 5.3 The CWT Framework

The initial basis for a "Complex Wavelet Tracker Framework" is introduced. The framework incorporates background stabilization, moving target detection, aided track gate initialization, and prediction algorithms which are based on the Complex Wavelet Tracker. An overview of the framework is given next which is followed by the presentation of the blocks in the subsequent sections.

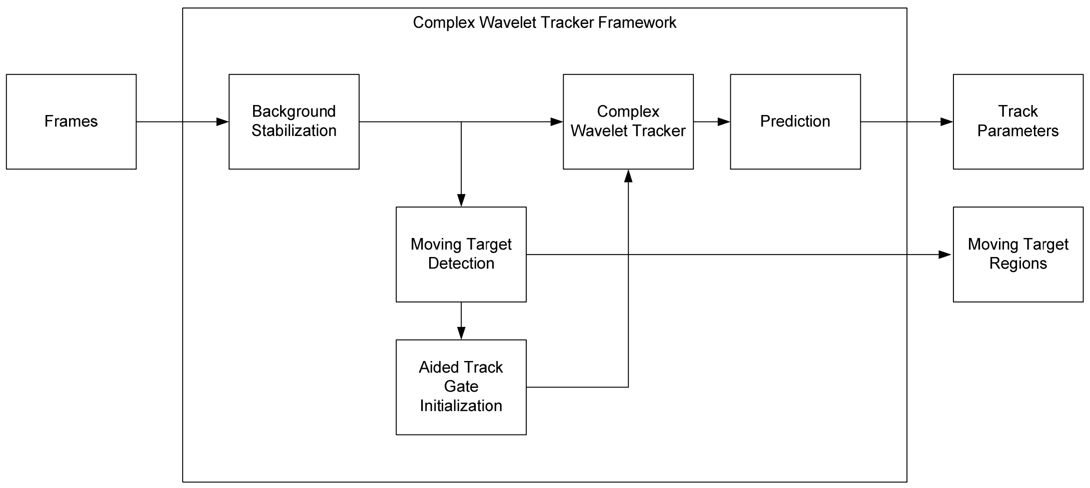


**Figure 5.2** The structure of the Complex Wavelet Tracker Framework.

### 5.3.1 Overview of the Framework

The Complex Wavelet Tracker Framework consists of several *functional blocks*, which are *background stabilization*, *moving target detection*, *aided track gate initialization* and *track parameter prediction*. Background stabilization is needed to eliminate background motion from target motion. This is necessary in order to use a predictor, and background stabilization is required for the moving target detection and aiding track gate initialization blocks. Utilizing background stabilization also improves the performance of the tracker. All functional blocks, except the predictor block, are based on the Complex Wavelet Tracker. The structure of the framework is given in Figure 5.2.

### 5.3.2 Background Stabilization

Background stabilization is performed using the second method explained in Section 5.2.2 where a number of small gates are used. The solution is then obtained from these gates according to two motion models: translational and affine. For the tracker and the predictor, translational motion is sufficient. The affine solution is used for the moving target detection and track gate initialization blocks.

Background motion is extracted from ten (10) fixed track gates that are located near the frame boundary. A sample placement of the regions is shown in Figure 5.1. The motion of each gate is computed using the Complex Wavelet Tracker.

The *translational* motion model is simpler and is sufficient to stabilize the background for the predictor and tracker blocks. The displacement is obtained using least squares estimation. The estimation is performed at every frame.

The *translational* motion model is not accurate enough for the detection of moving objects. Therefore the *affine* motion model is used for this purpose. The affine parameters are estimated using least squares method. Affine estimation can be performed at a lower pace, like every five or ten frames, depending on the dynamics of the scene.

### 5.3.3 Moving Target Detection

Moving targets are detected using the *affine* motion model in the background stabilization block. The second frame is warped according to the affine parameters and is subtracted from the reference frame to obtain a frame difference. The moving regions could then be extracted using some other method. The computation is performed at a lower pace like every five or ten frames in order to allow for sufficient motion between frames.

Any detection scheme that is suitable could be used for the estimation of moving objects for this block. For the scope of this thesis, only the alignment of the frames is computed in order to be used in frame differencing.

### 5.3.4 Aided Track Gate Initialization

A track gate could be opened automatically, manually or semi-automatically. In the automatic mode, a track gate is initiated automatically on the moving objects that are detected by the moving target detection block. In the manual mode, the track gates are initiated manually.

The semi-automatic mode, which we also call aided track gated initialization, operates in two ways. The moving targets are still detected and displayed to the operator, but the tracker is not activated. The first way is to manually activate the tracker by selecting the highlighted track gate.

The other way is necessary for occasions when the target could not be detected by the moving target detector. For this case, the operator simply points to the target without opening a track gate. The track gate initialization block searches for a target starting from this point and opens a gate around the segmented region. This block should be capable of extracting both moving and stable targets.

In this way, easy track gate initialization is performed. This is important in defense applications where the platform is not stable and the target is moving. It is difficult in such conditions to manually open a track gate on the targets. This algorithm is aimed to aid the operator in such situations.

### 5.3.5 Prediction

The prediction block is used to filter and estimate the track parameters in the absence of observations throughout the tracking. Prediction is performed on the track parameters according to previous estimates and frame observations.

In order to use a predictor successfully, platform data together with relative camera motion (*azimuth* and *elevation*) are required. In the absence of this data, this information should be extracted from the frames as well. The first step is to separate target motion from background motion. Background motion is mainly caused due to ego-motion, which is the motion of the camera and/or platform itself. This can be accomplished using the background stabilization block. Finally, the stabilized target motion can be used by the predictor.

Any prediction algorithm suitable for the specific application can be used. The Kalman [45] or the CONDENSATION [44] filter are among many other prediction algorithms.

# CHAPTER 6

# RESULTS

This Chapter presents the simulations performed for the evaluation of the proposed algorithm. The first simulations are related with the motion estimation performance of the original CDWT-ME algorithm of Magarey and Kingsbury. A summary of the results are provided in the first section.

After these preliminary investigations, quantitative evaluations are performed on the proposed algorithm to obtain quantitative results on the accuracy, robustness, and limits of the Complex Wavelet Tracker. The results are compared with Lucas-Kanade. The second section presents these quantitative results.

These evaluations are followed by qualitative simulations to test the performance of the proposed method on real sequences. A number of real sequences have been used to evaluate the tracking performance on different scenarios using different video sources like infrared and day video. The results are given in the last section.

## 6.1 Preliminary Tests: Motion Estimation

In this section the motion estimation performance of the original CDWT-ME algorithm of Magarey and Kingsbury [60] will be compared to Lucas-Kanade [57] and Horn-Schunck [42] motion estimation algorithms. This evaluation is important since the proposed method is based on the original CDWT-ME algorithm. As both methods depend upon the phase of the complex wavelet transform, this initial evaluation will present the potential performance of the algorithm. In other words, to understand the adequacy of the CDWT in determining moving pixels or regions and its ability to relate them among successive frames is important for the success of the proposed tracking algorithm.

For the evaluation of the performance of the dense flow field generated by the CDWT-ME algorithm, we used the Lucas-Kanade [57] and Horn-Schunck [42] algorithms for comparison. These algorithms have been chosen according to the evaluations performed by Barron *et al* [5] on a wide range of optical flow estimation methods. In his work in [5], Barron *et al* observed that the most successful and precise algorithm were the phase-based algorithm of Fleet and Jepson [30]. Then, the next two were Lucas-Kanade and Horn-Schunck algorithms. Since the CDWT-ME is an implementation of Fleet and Jepson's algorithm, we chose the next two for comparison.

The performance of both algorithms is based mainly on the computation of the gradients. Hence, different implementations can yield different results. Therefore, among several implementations of Lucas-Kanade and Horn-Schunck algorithms we observed that the implementation of Barron [6] resulted in the best performance. Hence, we reused this implementation. In this implementation, 5 frames are used to compute the gradients. For most cases, prior to the gradient computation, a 3-D Gaussian filter with sigma = 1.5 is used, which requires the use of 15 frames.

In the tests, we included both thresholded and full versions of the results for all three algorithms in order to evaluate the performance of the methods at low confident pixels. For the CDWT-ME, we used the confidence measure defined in Equation (5.8) over a single pixel; and for the other two methods, the threshold *Tau* [5] is used. For *Tau* = 0.0, no thresholding is applied and the flow field and errors are computed for the whole image.

The next Section will present the error measures. A summary and discussion on the results is presented afterwards. The test sequences used in the evaluations together with the test results are given in Appendix A.

### 6.1.1 Error Measures for Motion Estimation

Two error measurements have been used, one for angular error and the other for magnitude error. The angular error measure [5] is defined by

$$e_\theta = \arccos\left( \frac{\hat{d}_1 d_1 + \hat{d}_2 d_2 + 1}{\sqrt{\hat{d}_1^2 + \hat{d}_2^2 + 1}\sqrt{d_1^2 + d_2^2 + 1}} \right) \tag{6.1}$$

and the magnitude measure is defined by

$$e_M = \left| \sqrt{\hat{d}_1^2 + \hat{d}_2^2} - \sqrt{d_1^2 + d_2^2} \right| \tag{6.2}$$

where $\mathbf{d}(\mathbf{x}) = [d_1(x_1, x_2), d_2(x_1, x_2)]^T$ is the correct flow vector and $\hat{\mathbf{d}}(\mathbf{x})$ is the estimated flow vector. For each image pair the mean and standard deviation of the angle and magnitude errors have been computed.

### 6.1.2 Summary of the Results

Simulations have been performed in order to examine the estimation performance of the CDWT-ME algorithm. The results have been compared with the results of the LK and HS methods. (See Appendix A for the results.)

For the Sinusoid sequences, the CDWT-ME has a lower performance than the LK and HS methods. However, for the Square sequence, it performed better and than the others. For the Translating and Diverging Tree Sequences the CDWT-ME was slight behind the LK and HS methods. For the Yosemite Sequence, all three methods produced similar error results. However, on the overall, the CDWT-ME was more reasonable especially on the clouds region of the sequence. Finally, for the four real sequences, the CDWT-ME produced a more reasonable flow field than the other methods.

To understand the advantages of the CDWT-ME to the other two methods we compared the results in a different way. Table 6.1 presents the results of the Translating and Diverging Tree Sequences and the Yosemite Sequence from another perspective. It gives the percentage of the number of pixels having errors within a certain range. The first column gives the percentage of the pixels for which a flow is

**Table 6.1** Error distributions for the Translating and Diverging Tree, and Yosemite Sequences.

| *Sequence* | *Algorithm* | *Density* | | *Angle Error Distribution (%)* | | | *Magnitude Error Distribution (Normalized) (%)* | | |
|---|---|---|---|---|---|---|---|---|---|
| | | *Total* | *%* | *≤ 1°* | *≤ 3°* | *≤ 5°* | *≤ 1%* | *≤ 3%* | *≤ 5%* |
| ***Yosemite*** | CDWT-ME (Conf. T. = 0.00) | 73888 | 92.8 | 7.9 | 36.3 | 55.2 | 6.1 | 17.7 | 28.1 |
| | CDWT-ME (Conf. T. = 0.95) | 39104 | 49.1 | 5.5 | 22.4 | 32.6 | 3.8 | 10.9 | 17 |
| | Lucas-Kanade (Tau = 1.0) | 25669 | 32.2 | 8.6 | 23.1 | 27.4 | 4.2 | 11.8 | 17.3 |
| | Lucas-Kanade (Tau = 0.0) | 64512 | 81.0 | 12.6 | 38.8 | 49.5 | 6.9 | 19.6 | 29.8 |
| | Horn-Schunck (Tau = 5.0) | 21216 | 26.6 | 6.6 | 16.6 | 20.3 | 3.2 | 9.0 | 13.1 |
| | Horn-Schunck (Tau = 0.0) | 64512 | 81.0 | 11.3 | 35.8 | 47.5 | 6.4 | 18.4 | 28.2 |
| | CDWT-ME (at LK Tau = 1.0) | 25554 | 32.1 | 3.6 | 15.4 | 22.8 | 2.7 | 7.7 | 11.8 |
| ***Translating Tree*** | CDWT-ME (Conf. T. = 0.00) | 20108 | 89.4 | 32 | 71.5 | 80.9 | 17.4 | 46.5 | 64.2 |
| | CDWT-ME (Conf. T. = 0.95) | 17528 | 77.9 | 30.8 | 65.9 | 73.3 | 16.3 | 43.5 | 59.4 |
| | Lucas-Kanade (Tau = 1.0) | 6845 | 30.4 | 24.6 | 30 | 30.4 | 15.2 | 27.4 | 29.6 |
| | Lucas-Kanade (Tau = 0.0) | 14884 | 66.2 | 44.3 | 62.6 | 65.1 | 24.2 | 49.7 | 58.5 |
| | Horn-Schunck (Tau = 5.0) | 7913 | 35.2 | 17 | 28.9 | 31.8 | 11.0 | 22.7 | 27.5 |
| | Horn-Schunck (Tau = 0.0) | 14884 | 66.2 | 27.8 | 52.5 | 59.9 | 16.9 | 38.5 | 48.6 |
| | CDWT-ME (at LK Tau = 1.0) | 6841 | 30.4 | 10.4 | 25.4 | 28.3 | 5.8 | 16.8 | 23.8 |
| ***Diverging Tree*** | CDWT-ME (Conf. T. = 0.00) | 19934 | 88.6 | 6.9 | 38.6 | 61.3 | 7.0 | 20.6 | 33.7 |
| | CDWT-ME (Conf. T. = 0.95) | 11479 | 51.0 | 5.0 | 26.5 | 39.9 | 4.6 | 13.7 | 22.3 |
| | Lucas-Kanade (Tau = 1.0) | 8090 | 36.0 | 10.9 | 30.2 | 34 | 7.6 | 18.8 | 25.5 |
| | Lucas-Kanade (Tau = 0.0) | 14884 | 66.2 | 15.0 | 48.9 | 59.6 | 11.7 | 29.6 | 41.5 |
| | Horn-Schunck (Tau = 5.0) | 7967 | 35.4 | 11.5 | 28.0 | 32.2 | 7.2 | 18.8 | 25.1 |
| | Horn-Schunck (Tau = 0.0) | 14884 | 66.2 | 17.9 | 51.5 | 60.7 | 11.7 | 31.8 | 43.7 |
| | CDWT-ME (at LK Tau = 1.0) | 8044 | 35.8 | 3.3 | 18.5 | 28.7 | 3.3 | 9.6 | 15.7 |

computed, the second column gives the percentage of angle errors smaller than 1°, 3°, and 5°, and the third column gives the percentage of normalized magnitude errors smaller than 1%, 3% and 5%.

For all three sequences, the CDWT-ME produced the highest number of flows. In most of the cases LK produced the most accurate results. The HS with no thresholding performed better in the Diverging Tree sequence in both angle and magnitude. The CDWT-ME performed the best for the angle error within 5° for all three sequences.

In the overall, the CDWT-ME produced reasonable flow fields for all the sequence. It also performed very successfully in low textured regions where the HS and LK algorithms performed poorly. This is important for our purpose as we require the tracking algorithm to track even under worse conditions. In addition to this, the CDWT-ME has produced highly more dense results than the other methods.

Another fact to note is that the CDWT-ME algorithm has produced these results using only 2 frames, whereas the LK and HS algorithms required 15 frames. This is important for tracking since it should run in real-time. Actually, the LK and HS methods could be implemented to run with 2 frames only, but, as we mentioned before, the performance of these methods rely highly on the estimation of the gradients and the implementation of Barron were the one yielding the best results among other implementations.

As a result, we can say that, even not as accurate as the LK and HS for some conditions, the CDWT-ME produced quite accurate results for all the sequence with reasonable flow fields even for difficult scenes.

## 6.2 Quantitative Tests

Quantitative Tests include the test where the true motion parameters are known. The test sequences have been created synthetically by deforming a real still image under various affine deformations. The purpose of these tests is to obtain a quantitative evaluation of the Complex Wavelet Tracker. Comparisons have been performed using the Lucas-Kanade [6] [57] method with the *affine* motion model. The implementation of Baker and Matthews [4] for the Lucas-Kanade method is used in the simulations.

The following tests have been performed for the evaluation of the proposed algorithm:

1. Tests for exploring affine estimation limits
2. Tests for understanding the effect of changing the number of levels used
3. Tests to compare the performance of the three motion models
4. Tests for evaluating the robustness to illumination changes
5. Tests for evaluating the performance under noise
6. Tests to understand the effect of the target intensity pattern
7. Tests to evaluate performance under random deformations and perturbations

Three types of motion models for the Complex Wavelet Tracker has been explored: the *translational* motion model, the *similarity* motion model, and the *affine* motion model. In addition to this, the effect of different target sizes and the role of the number of levels used have been examined. Finally, robustness tests have been performed in terms of illumination changes, noise, and texture dependency.

In order to quantify the results, error measures have been defined which are given in the following section. According to these measures, the error range is segmented into three quality criteria regions: (1) the "*precise*" tracking region, where the error is negligible; (2) the "*acceptable*" tracking region where the algorithm manages to maintain the track gate on the target, although not precise; (3) the "*lost*" region where

[6] The original version of the Lucas-Kanade [54] is an image registration method and the use of Lucas-Kanade for tracking is later proposed by Shi and Tomasi [79]. They extended this method by including the affine motion model and used it for tracking point features. This tracker is known as the "Lucas-Kanade Feature Tracker" or "KLT (Kanade-Lucas-Tomasi) Tracker".

In this work, however, we preferred to keep the original version of Lucas-Kanade [54] and used the affine motion model. This is consistent with the definition of tracking of Hager and Belhumeur [39]. Recently, Baker and Matthews [3] showed that Lucas-Kanade [54] and Hager-Belhumeur [39] are related. (See Section 2.2.1 for a review of these methods in the context of this work). This is the reason of choosing the original method of Lucas-Kanade [54] as an intensity-based approach in contrast to the proposed method, which is phase-based.

the target has been lost. These error levels for each criterion are defined in the next section (See Table 6.2).

The next section defines the error measures used for the quantitative evaluations. The test sequences that have been produced and used for the quantitative tests are presented next which is followed by a number of subsequent sections presenting the results of the quantitative tests.

### 6.2.1 Error Measures

Besides the definition of the *affine* motion model in Equation (4.3), a rearrangement of terms will yield a more known definition of the *affine* motion model as follows:

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}. \tag{6.3}$$

If the present position from **A** is taken out, we obtain

$$\mathbf{x}' = (\mathbf{1} + \mathbf{D})\mathbf{x} + \mathbf{b} = \mathbf{x} + (\mathbf{D}\mathbf{x} + \mathbf{b}) \tag{6.4}$$

where $\mathbf{A} = \mathbf{1} + \mathbf{D}$ and the elements of **D** are equal to the first four parameters $p_1 - p_4$. **D** defines the affine portion, namely, *rotation*, *shear* and *scale* type of deformations, whereas **b** corresponds to the last two parameters $p_5$ and $p_6$ and gives the *translation* portion.

In order to have a compact representation of the errors, we combine the six parameters into two terms, namely the **D** and **b**, corresponding to the *affine* and *translational* components of motion. This also helps us in collecting the same type of parameters on the same term, since the order of magnitude of the errors for the *affine* and *translation* parameters are different.

The relation between the parameters and **A** and **b** for each motion model is as follows:

*Affine Motion Model:*
$$\mathbf{A} = \begin{bmatrix} 1+p_1 & p_3 \\ p_2 & 1+p_4 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} p_5 \\ p_6 \end{bmatrix} \tag{6.5}$$

*Similarity Motion Model:*
$$\mathbf{A} = \begin{bmatrix} 1+p_1 & -p_2 \\ p_2 & 1+p_1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} p_3 \\ p_4 \end{bmatrix} \tag{6.6}$$

*Translational Motion Model:* $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\mathbf{b} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$ (6.7)

Then, for the error in the *affine* parameters, **A**, the following measure [59] will be used:

$$e_{\mathbf{A}} = \left\| \mathbf{1} - \mathbf{A}\hat{\mathbf{A}}^{-1} \right\| \tag{6.8}$$

where ||…|| is the $L_2$ norm. Here, **A** is used instead of **D** and **1** is subtracted explicitly. Similarly, for the parameters $p_5$ and $p_6$ the error measure will be

$$e_{\mathbf{b}} = \left\| \hat{\mathbf{b}} - \mathbf{b} \right\|. \tag{6.9}$$

In these measures, $\hat{\mathbf{A}}$ and $\hat{\mathbf{b}}$ denote the *estimated* parameters, whereas **A** and **b** denote the *true* parameters. For the evaluations, both *instantaneous* and *cumulative* parameter errors are computed. *Instantaneous* parameters are estimated from two consecutive frames, whereas *cumulative* ones are accumulated from the very first frame. The *cumulative* parameters define the motion with respect to the track gate in the first frame of the tracking sequence.

A *compositional* approach [3] is used to obtain the cumulative parameters. Based on the *affine* motion model given in Equation (6.3), the cumulative parameters are computed as follows:

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1 \\ \mathbf{x}_2 &= \mathbf{A}_2\mathbf{x}_1 + \mathbf{b}_2 \\ &= \mathbf{A}_2(\mathbf{A}_1\mathbf{x}_0 + \mathbf{b}_1) + \mathbf{b}_2 \\ &= (\mathbf{A}_2\mathbf{A}_1)\mathbf{x}_0 + (\mathbf{A}_2\mathbf{b}_1 + \mathbf{b}_2) \\ &\vdots \end{aligned} \tag{6.10}$$

which yields

$$\begin{aligned} \overline{\mathbf{A}}_n &= \mathbf{A}_n\overline{\mathbf{A}}_{n-1} \\ \overline{\mathbf{b}}_n &= \mathbf{A}_n\overline{\mathbf{b}}_{n-1} + \mathbf{b}_n \end{aligned} \tag{6.11}$$

where $\overline{\mathbf{A}}_0 = \mathbf{1}$ and $\overline{\mathbf{b}}_0 = \mathbf{0}$. Here, $\overline{\mathbf{A}}$ and $\overline{\mathbf{b}}$ denote the cumulative parameters.

Technically, the parameters **A** and **b** are not affected from the origin of the coordinate system, independently. However, the origin of an affine motion directly affects the translational parameter **b**. Since, all types of affine motion like rotation and scale are with respect to the origin of the coordinate system. If the origin of the affine motion is not at the origin of the coordinate system, which is the case in all our test sequences, the translational parameter **b** compensates for this difference, yielding larger values. So, in order to eliminate this, the origin of the coordinate frame is shifted to the origin of the track gate at each frame for the error computations.

Based on these, the following error measures have been used for performance evaluations:

*Instantaneous Error in* **A** *for the* $n^{th}$ *frame:*

$$e_{\mathbf{A}} = \left\| \mathbf{1} - \mathbf{A}_n \hat{\mathbf{A}}_n^{-1} \right\| \tag{6.12}$$

*Instantaneous Error in* **b** *for the* $n^{th}$ *frame:*

$$e_{\mathbf{b}} = \left\| \hat{\mathbf{b}}_n - \mathbf{b}_n \right\| \tag{6.13}$$

*Average Instantaneous Error for* **A** *over N frames:*

$$e_{\tilde{\mathbf{A}}} = \frac{1}{N} \sum_{n=1}^{N} \left\| \hat{\mathbf{A}}_n - \mathbf{A}_n \right\| \tag{6.14}$$

*Average Instantaneous Error for* **b** *over N frames:*

$$e_{\tilde{\mathbf{b}}} = \frac{1}{N} \sum_{n=1}^{N} \left\| \hat{\mathbf{b}}_n - \mathbf{b}_n \right\| \tag{6.15}$$

*Error in cumulative* **A** *for the* $n^{th}$ *frame:*

$$e_{\bar{\mathbf{A}}} = \left\| \hat{\bar{\mathbf{A}}}_n - \bar{\mathbf{A}}_n \right\| \tag{6.16}$$

*Error in cumulative* **b** *for the* $n^{th}$ *frame:*

$$e_{\bar{\mathbf{b}}} = \left\| \hat{\bar{\mathbf{b}}}_n - \bar{\mathbf{b}}_n \right\| \tag{6.17}$$

*Error in Track Gate for the n^th frame:*

$$e_G = \frac{1}{4}\left[\left\|\hat{\mathbf{g}}_n^{(UL)} - \mathbf{g}_n^{(UL)}\right\| + \left\|\hat{\mathbf{g}}_n^{(UR)} - \mathbf{g}_n^{(UR)}\right\| + \left\|\hat{\mathbf{g}}_n^{(LL)} - \mathbf{g}_n^{(LL)}\right\| + \left\|\hat{\mathbf{g}}_n^{(LR)} - \mathbf{g}_n^{(LR)}\right\|\right] \qquad (6.18)$$

Note that in the last equation the track gate is defined by its four corners: *upper-left* (UL), *upper-right* (UR), *lower-left* (LL), and *lower-right* (LR), which are represented with $\mathbf{g}^{(\mathrm{UL})}$, $\mathbf{g}^{(\mathrm{UR})}$, $\mathbf{g}^{(\mathrm{LL})}$, and $\mathbf{g}^{(\mathrm{LR})}$, respectively. Then, the error in the track gate is defined by the average of the distance of the corner points.

Referring to the criteria for the track quality defined in the previous section, we will quantify the regions using the error measures of **A** and **b** defined in Equations (6.8) and (6.9), respectively, as given in Table 6.2.

**Table 6.2** Quantitative limits of parameters **A** and **b** for tracking criteria.

| *Criteria* | *Limits for* **A** | *Limits for* **b** |
|---|---|---|
| Precise | $e_{\mathbf{A}} \leq 0.005$ | $e_{\mathbf{b}} \leq 0.05$ |
| Acceptable | $0.005 \leq e_{\mathbf{A}} \leq 0.05$ | $0.05 \leq e_{\mathbf{b}} \leq 0.5$ |
| Lost | $0.05 \leq e_{\mathbf{A}}$ | $0.5 \leq e_{\mathbf{b}}$ |

When the limits for **A** and **b** conflict for some cases, we will decide in favor of **A**, i.e. if **A** is within limits, and **b** is slightly out of limit, then we will choose the criteria where **A** fits in. These limits roughly mean the following: For **A**, 0.005 corresponds to 0.5% of scaling, 0.5% of shearing or about 0.3° of rotation; 0.05 corresponds to 5% of scaling, 5% of shearing, or about 3° of rotation. For **b**, the value directly converts to frame coordinates. So, 0.05 refers to 0.05 pixels of difference and 0.5 refers to a half pixel error. So, these limits will be used to obtain the practical limits for the Complex Wavelet Tracker in the subsequent sections.

### 6.2.2 Test Sequences

Test sequences have been created to measure the performance of the proposed algorithm. These sequences have been created by deforming still images under controlled affine deformations. We used three different still images: Lenna, Grass, and Cloud, shown in Figure 6.1, each of which has its own properties and represent different difficulty levels for motion estimation.

The famous “Lenna” image represents a nice texture where the intensity levels are covering the full range. The “Grass” image is created to represent a random-like pattern that is similar and monotonic over the whole image. The “Cloud” image is created to represent a smooth low-contrast surface which is especially difficult to track under illumination changes. All images are 256 × 256 pixels in size. The range of the intensity values are 0 – 255.

The sequences are created for the four basic motion types: translation, scale, shear, and rotation. For each motion type, a number of sequences have been produced for different degrees of the specific deformation.

The *affine* motion model defined in Equation (6.3) is used to deform the *reference* image. The affine parameters **A** and **b** are computed for each sequence. Each frame in the sequence is warped according to the *cumulative* affine parameters (6.11) with respect to the first frame using *bilinear* interpolation.

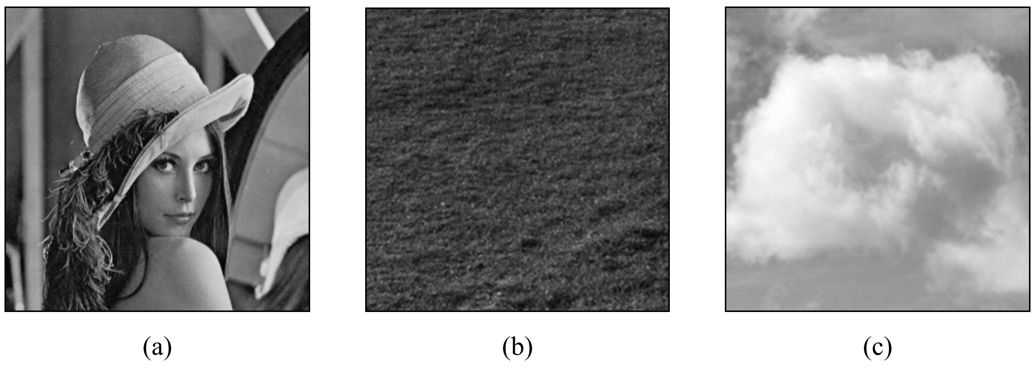


**Figure 6.1** Still images used to create the test sequences. (a) The famous “Lenna” image. (b) The “Grass” image, showing repetitive similar monotonic texture. (c) The “Cloud” image, showing a very soft texture having very low contrast.

For **translation**, 24 test sequences have been created for displacements ranging from 0.25 pixels to 12.00 pixels with a step of 0.25 pixels for the range up to 4.00 pixes and a step of 1.00 pixels thereafter. All the translating sequences are 25 frames long.

In order to keep the track gate within frame limits, the sequence has been extended to 512 × 256 pixels. A sample frame is shown in Figure 6.2 (b) where we clipped the frame to its original size for demonstration purposes.

For **scaling**, 40 test sequences have been created for scale ratios ranging from –20% to +20% with a step of 1%. All of the scaling sequences are 20 frames long. In order to keep the target within frame borders, the sequence for scaling up is extended to 512 × 512 pixels. A sample frames is shown in Figure 6.2 (d) and (e) where the frame for the up scaling sequence is clipped to its original size for demonstration purposes.

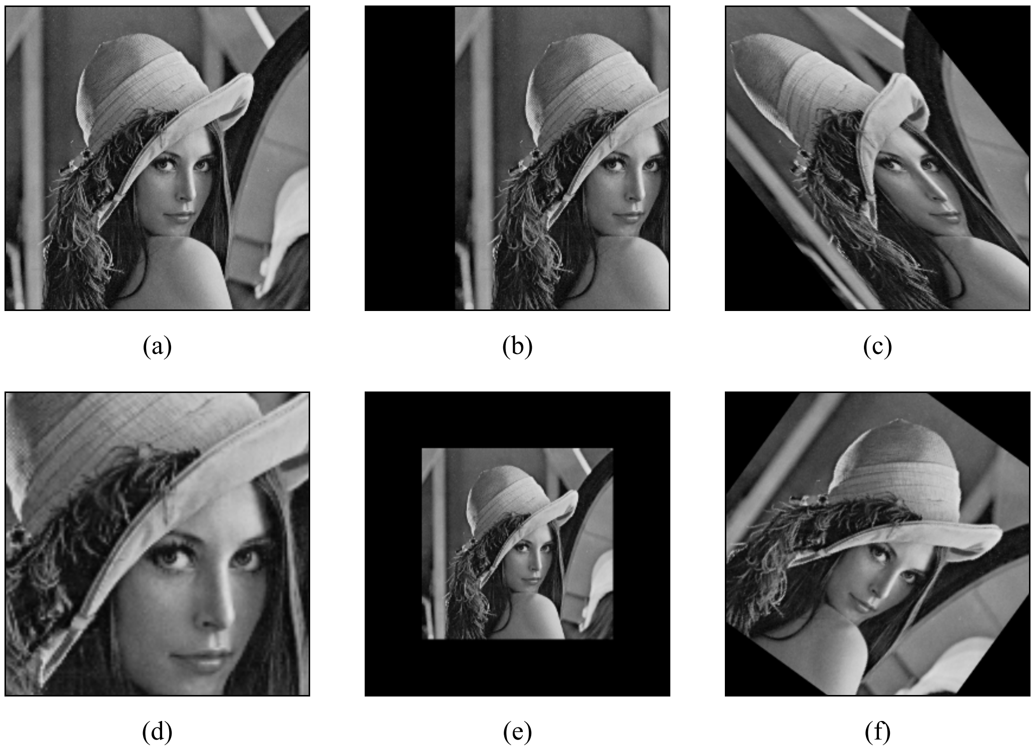


**Figure 6.2** Test sequences for affine deformations. (a) First frame of the Lenna Sequences. (b) Translation. (c) Shear. (d) Up scaling. (e) Down scaling. (f) Rotation.

For **shear**, 20 test sequences have been created for shear ratio ranging from 1% to 20% with a step pf 1%. All of the shear sequences are 30 frames long. In order to keep the track gate within frame limits, the sequence has been extended to 512 × 256 pixels. A sample frame is shown in Figure 6.2 (c) where we clipped the frame to its original size for demonstration purposes.

For **rotation**, 20 test sequences have been created for rotation angles ranging from 1° to 10° with a step of 1°. All of the rotation sequences are 360 frames long. A sample frame is shown in Figure 6.2 (f).

Perturbations have also been added for the robustness tests. Three types of perturbations are used: *brightness* change, *contrast* change, and *noise*. For brightness, an offset is added to the pixel intensity values. For contrast, the pixel intensity values are scaled. The additive offset is varied from 0 to 25, and the scaling factor from 0.75 to 1.25. For noise, additive Gaussian noise is added to each pixel intensity value. Noise standard deviation is varied from 0 to 20 where the mean is set to 0. For all perturbations, clipping is performed on saturating entries to prevent over- and underflows of data.

A total of 1.434 Test Sequences have been created and 15.774 simulations have been performed. The subsequent sections will present the results.

### 6.2.3 Test 1: Exploring Affine Limits – Effect of Gate Size

The purpose of this test is to explore the practical limits of the Complex Wavelet Tracker to the following types of deformations: translation, scale, shear, and rotation. These practical limits are compared to the theoretical ones presented in Section 4.2 at the end of this section.

The Lenna Test Sequences presented in Section 6.2.2 are used. For the measurement of errors, the average instantaneous errors for the parameters **A** and **b**, namely (6.14) and (6.15), are used. The *affine* motion model is used to perform these tests. The results of each type of motion are given in the following subsections.

The simulations have been performed for three different target sizes: 32 × 32 pixels, 64 × 64 pixels, and 160 × 160 pixels, considered as "small", "medium" and "large" size targets, respectively. These are the same sizes used in the theoretical calculations in Section 4.2.

#### 6.2.3.1 Translation

The Translating Lenna Test Sequences are used for the simulations. All of the 25 frames are covered in the error calculations. The results are shown in Figure 6.3 and Figure 6.4.

For a target size of 32 × 32 pixels, the error starts increasing slightly after 5 pixel displacement and breaks down at 8 pixels. For a target size of 64 × 64 pixels, very accurate tracking is performed up to 9 pixels, which is slightly above the theoretical limit (which is 8 pixels). The track breaks down after 10 pixels. For a target size of 160 × 160 pixels, even a translation of 11 pixels is estimated with very high precision and the algorithm breaks down thereafter.

As a result, for translation, the larger the target size, the more accurate the results are. For translations up to 5 pixels, the average instantaneous errors in **A** are around 0.0025 for 32 × 32 pixels, 0.0015 for 64 × 64 pixels, and 0.0002 for 160 × 160 pixels target size. This result shows that by increasing the support and incorporating more pixels into the solution, the accuracy of the algorithm increases. Moreover, the estimation limits exceed the theoretical ones, which is also a positive outcome of the increase in the support.

#### 6.2.3.2 Scale

The Scaling Lenna Test Sequences are used for the simulations. The average instantaneous errors are computed over the first five frames. The results are depicted in Figure 6.5 and Figure 6.6.

For 32 × 32 and 64 × 64 pixels target sizes, the accuracy remained within the acceptable range in between ±20% scale ratios where the scaling tests were performed. The estimates for the 64 × 64 pixel target were accurate in the range from −9% to +8%, whereas, for the 32 × 32 pixel target, it were accurate only in between −1% to +2%. For the 160 × 160 pixel target, however, the estimates were accurate from −12% to +10%, and acceptable for −16% to +12%.

On the overall, it is observed that, as the target size increases, the practical limits tend to exceed theoretical ones. This is the benefit of increasing the support. In addition to this, by comparing the accuracies at lower ratios, it is observed clearly that the accuracy of the algorithm increases as well with increasing target size, which is also an outcome of the increasing support.

#### 6.2.3.3 Shear

The Shearing Lenna Test Sequences are used for the simulations. The errors are computed over the first ten frames. The results are shown in Figure 6.7 and Figure 6.8.

The results for all target sizes are very accurate. For target sizes of 32 × 32 and 64 × 64 pixels, accurate estimation of the parameters have been obtained for up to 20% shear ratios. For the 160 × 160 pixel target the algorithm remained in the accurate region up to 16% shear ratio and gave acceptable results for 17%, but broke down after that.

On the overall, a clear increase in the accuracy of the algorithm is observed with increasing target size. The error in parameters $p_1 - p_4$ is around 0.0001 for a 160 × 160 pixel target for shear ratios up to 10%. The increase in the support has caused an increase in the accuracy of the estimation as more pixels contribute to the solution. For large target, the practical estimation limit has exceeded the theoretical one, which is also a benefit of the increase in the support.

#### 6.2.3.4 Rotation

The Rotating Lenna Test Sequences are used for the simulations. The first 25 frames are used in the error calculations. The results are depicted in Figure 6.9 and Figure 6.10.

Accurate results are obtained for rotation angles up to 6°, 10°, and 7° for target sizes 32 × 32, 64 × 64, and 160 × 160 pixels, respectively. Acceptable results are obtained for 32 × 32 pixels thereafter; and for 160 × 160 pixels target, acceptable tracking is performed until 8° and the track breaks down with 9° of rotation.

On the overall, by comparing the errors, it is observed that the accuracy of the estimation increases with an increase in target size. This is the positive effect of incorporating more pixels in the solution. Hence, for large targets, the theoretical limit of 5° rotation is exceeded, whereas, for the medium and small targets, the practical limit remained far below the theoretical ones.

#### 6.2.3.5 Summary of the Results

Simulations have been performed in order to explore the practical deformation limits of the proposed algorithm for different target sizes. The summary of the practical limits in comparison to the theoretical ones are collected in Table 6.3. The values in the tables are computed according to the definitions given in Section 4.2. The calculations are based on a maximum 8-pixel displacement for the 4-level pyramid.

**Table 6.3** Practical deformation limits for representative object sizes when a 4-level CDWT pyramid is used. Theoretically, a 4-level pyramid corresponds to a maximum displacement of 8 pixels. The values inside the parentheses are theoretical limits given in Table 4.1 of Section 4.2. The meanings of the values are explained in 4.2.

| *Object Size* | *Translation* | *Scale* | *Rotation* | *Shear* |
|---|---|---|---|---|
| 32 × 32 pixels | 25% (25%) | >20% (50%) | 6° (29°) | >20% (25%) |
| 64 × 64 pixels | 16% (12.5%) | >20% (25%) | 10° (14°) | >20% (12.5%) |
| 160 × 160 pixels | 7% (5%) | 12% (10%) | 7° (5°) | 17% (5%) |

For translation and shear type of motions, the theoretical limits have been met. For scale and rotation types, however, these limits were met only for large target sizes. For medium and small targets, the performance remained below the theoretical limits.

The reason for this is related with the fact that the CDWT has been designed according to the local translational model [59], which means that it is approximately invariant for translations only. Hence, for affine deformations, it is not invariant in the first place. However, increasing the support and incorporating a set of pixels instead of a single one extends the algorithm to handle affine deformations to an acceptable degree. Note that, although below the theoretical ones, the limits are actually quite high from an applicational point of view. For tracking, the application will run on real-time video sequences at rates corresponding to 33 msec (30 Hz) or 40 msec (25 Hz) time intervals between consecutive frames.

For all motion types, it is observed that the accuracy is increasing with an increase in the target size. The increase in the target size means that more pixels contribute to the solution, and hence, a better estimation is obtained. This increase in the support also enables the practical limits to exceed theoretical ones. This is observed for all motion types, as the limits for medium and small targets remained below the theoretical limits.

As a result, the accuracy of the algorithm increases with an increase in the target size due to an increase in the support leading to the correct solution. The maximum pixel displacement that can be handled does also increase with the increase in the support.

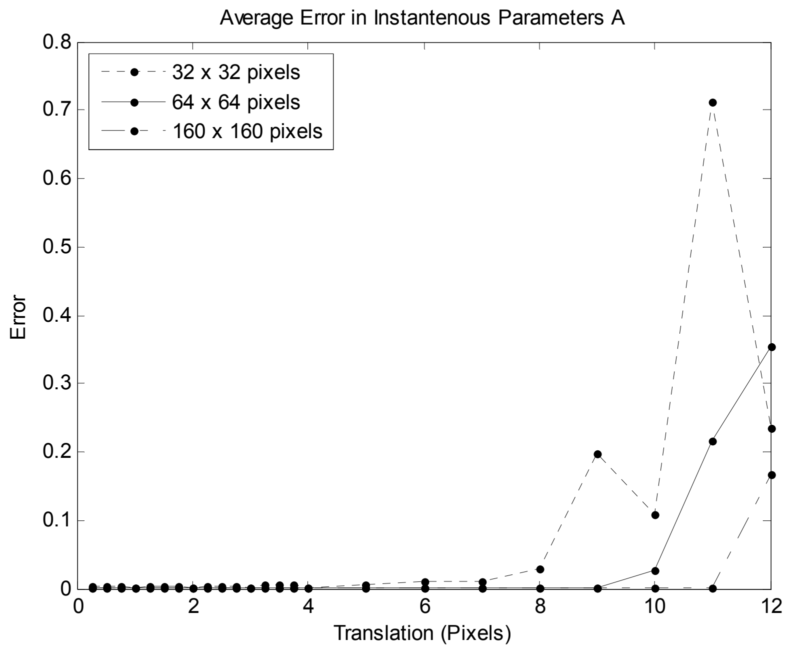


**Figure 6.3** Test 1 – Translation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Translating Lenna Test sequence for different target sizes. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

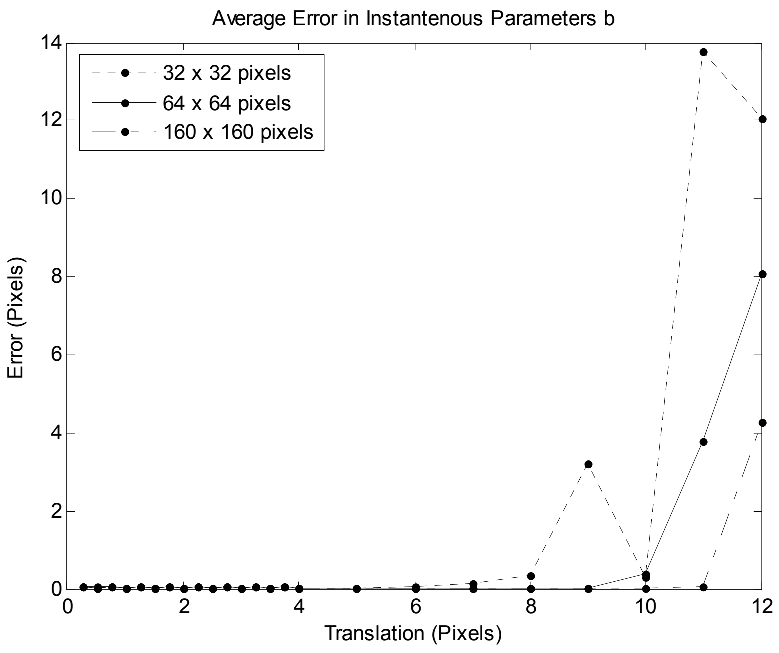


**Figure 6.4** Test 1 – Translation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Translating Lenna Test sequence for different target sizes. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

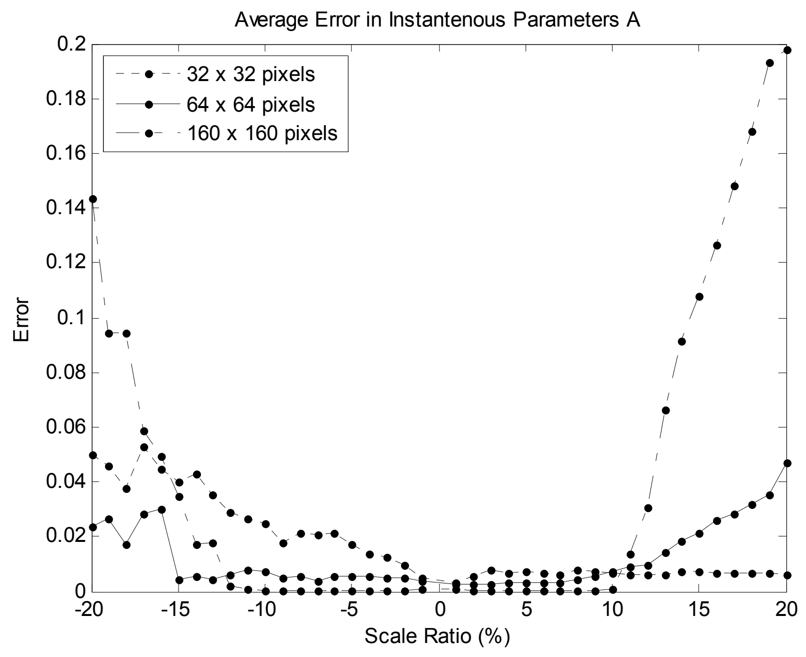


**Figure 6.5** Test 1 – Scale (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Scaling Lenna Test sequence for different target sizes. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

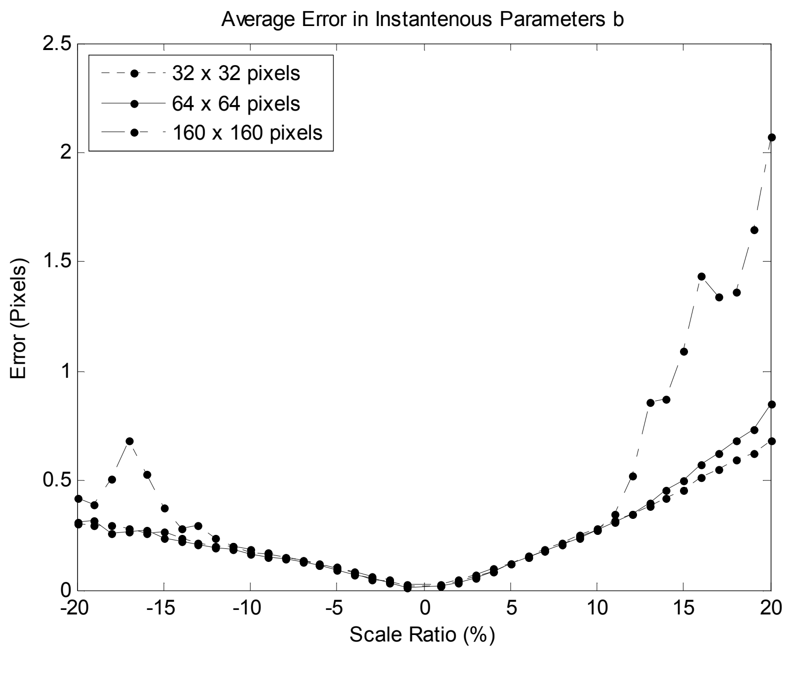


**Figure 6.6** Test 1 – Scale (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Scaling Lenna Test sequence for different target sizes. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

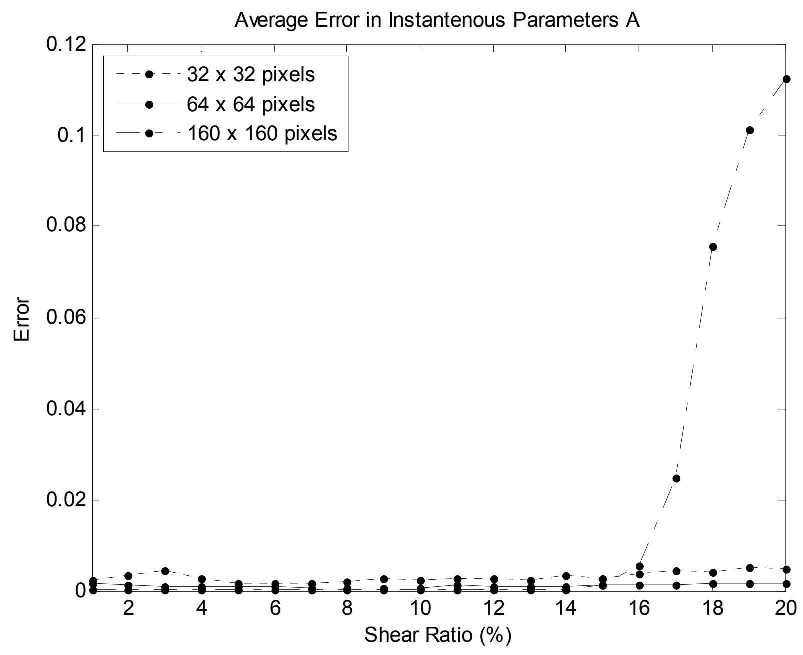


**Figure 6.7** Test 1 – Shear (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Shearing Lenna Test sequence for different target sizes. The errors are computed for shear ratios 1% – 20%. The average is computed over 10 frames.

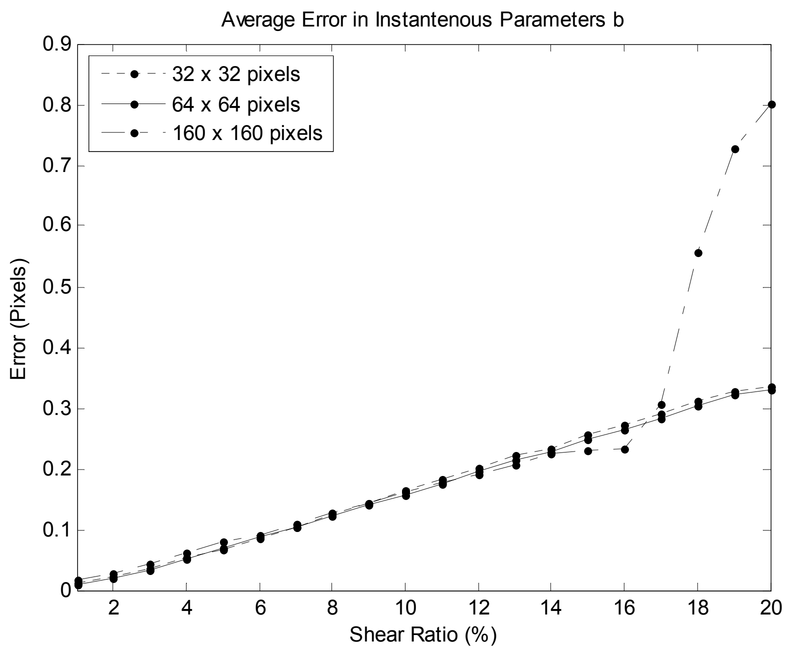


**Figure 6.8** Test 1 – Shear (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Shearing Lenna Test sequence for different target sizes. The errors are computed for shear ratios 1% –20%. The average is computed over 10 frames.

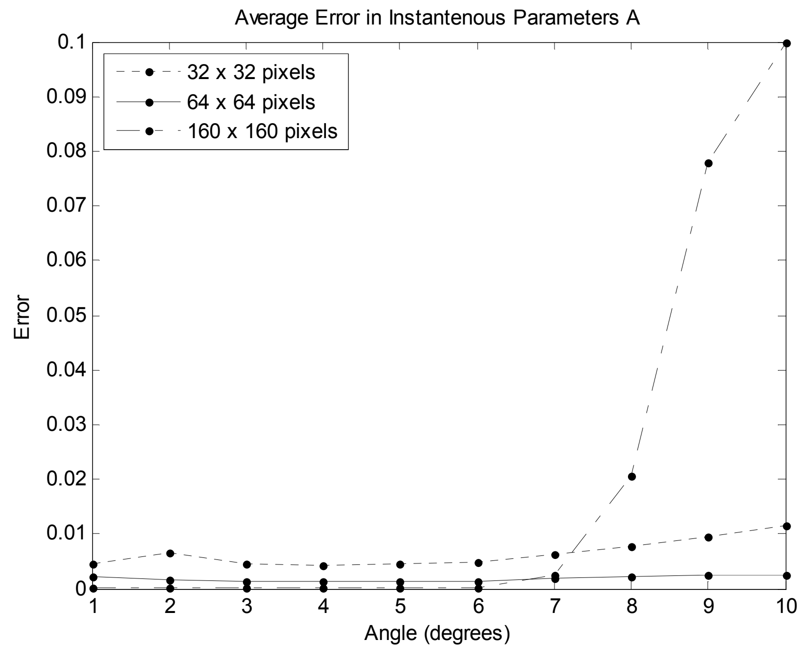


**Figure 6.9** Test 1 – Rotation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Rotating Lenna Test sequence for different target sizes. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

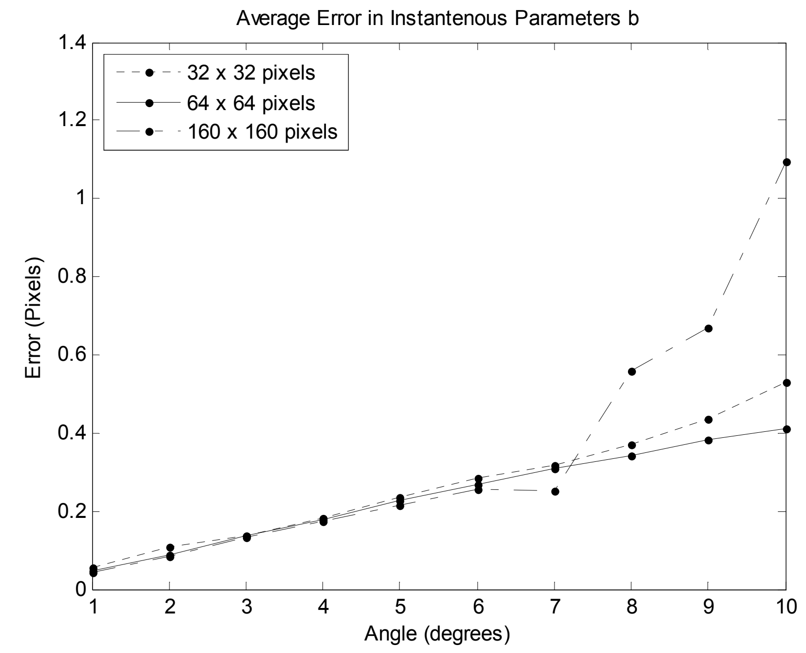


**Figure 6.10** Test 1 – Rotation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Rotating Lenna Test sequence for different target sizes. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

### 6.2.4 Test 2: Exploring Affine Limits – Effect of Number of Levels

These tests are performed to investigate the effect of the number of levels ($m_{\max}$) to the affine limits investigated in Test 1. The number of levels is varied from 3 to 6. The estimations are performed up to the first the level ($m_{\min} = 1$) for all tests. So, the number of levels in the CDWT pyramid will be equal to $m_{\max}$.

The Lenna Test Sequences are used with the 160 × 160 pixels size target. For the measurement of errors, the average instantaneous errors for the parameters **A** and **b** given by expressions (6.14) and (6.15) are used. For scaling, however, the instantaneous error measures (6.12) and (6.13) are preferred. The *affine* motion model is used in the tests. The results of each type of motion are given from Figure 6.11 to Figure 6.18.

For **translation**, when 3 levels are used, the track breaks down after 5 pixels whereas for 4 levels, the limit is 11 pixels. If the level is increased to 5 and 6, it is observed that the limit exceeds 12 pixels with an acceptable performance. Hence, a clear increase in the limits is observed with an increase in the number of levels used.

For **scale**, with an increase in the levels, a clear improvement is observed as well, especially in the affine parameter **A**. For 3 levels, a break occurs after ±7%, and for 4 levels, after around ±12%. For 5 and 6 levels, this break exceeds ±20%, leading to a clear improvement in the limits for scaling.

For **shear**, again, an improvement with an increase in the number of levels is observed. For this case, the track breaks after 8% with 3 levels, and 15% with 4 levels. With 5 and 6 levels, the limits go beyond 20%.

For **rotation**, the same improved is observed as well. With 3 levels, tracking is performed accurately until 4°, and with 4 levels up to 8°. With 5 and 6 levels, the limits exceed 10°.

As a summary, it is observed clearly that by increasing the number of levels ($m_{\max}$) the estimation limits have increased for all motion types, which is consistent with the theoretical outcomes. Furthermore, a very slight improvement in the estimation accuracy is observed as well.

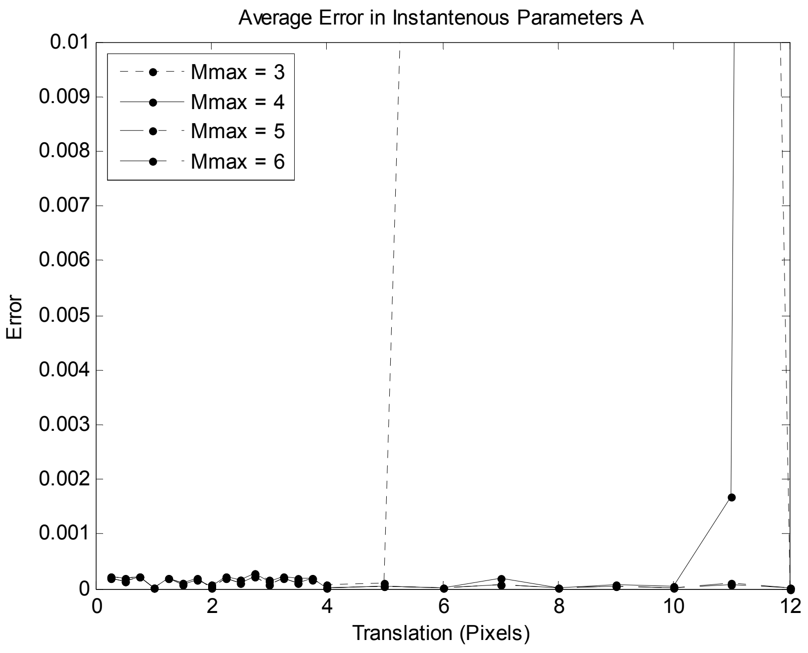


**Figure 6.11** Test 2 – Translation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Translating Lenna Test sequence for different number of levels. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

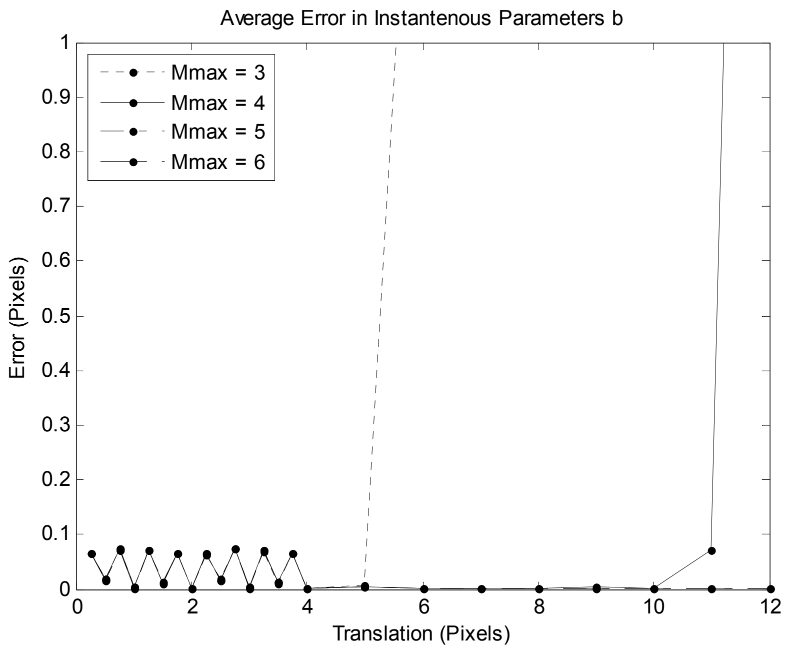


**Figure 6.12** Test 2 – Translation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Translating Lenna Test sequence for different number of levels. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

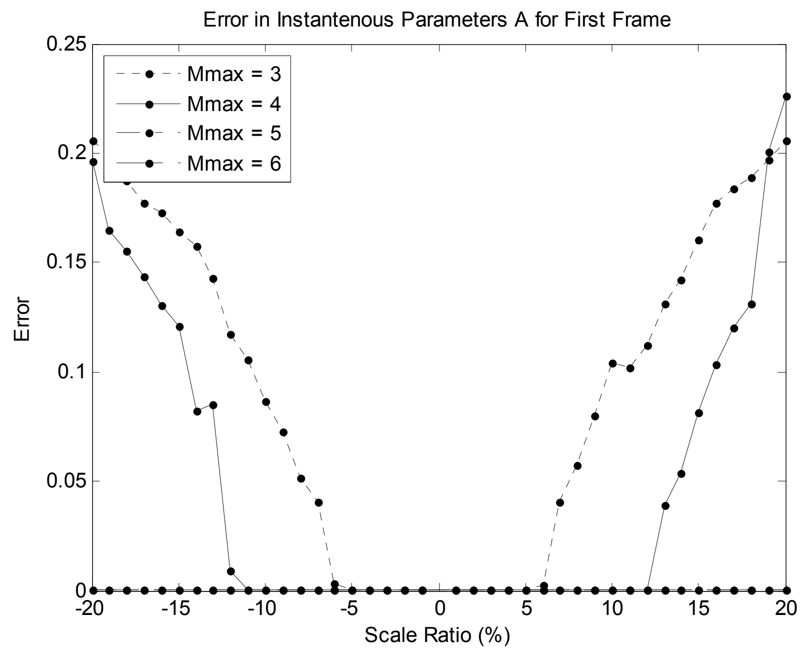


**Figure 6.13** Test 2 – Scale (**A**): Error in the instantaneous parameters $p_1 - p_4$ for the Scaling Lenna Test sequence for different number of levels. The errors are computed for scale ratios -20% – +20%.

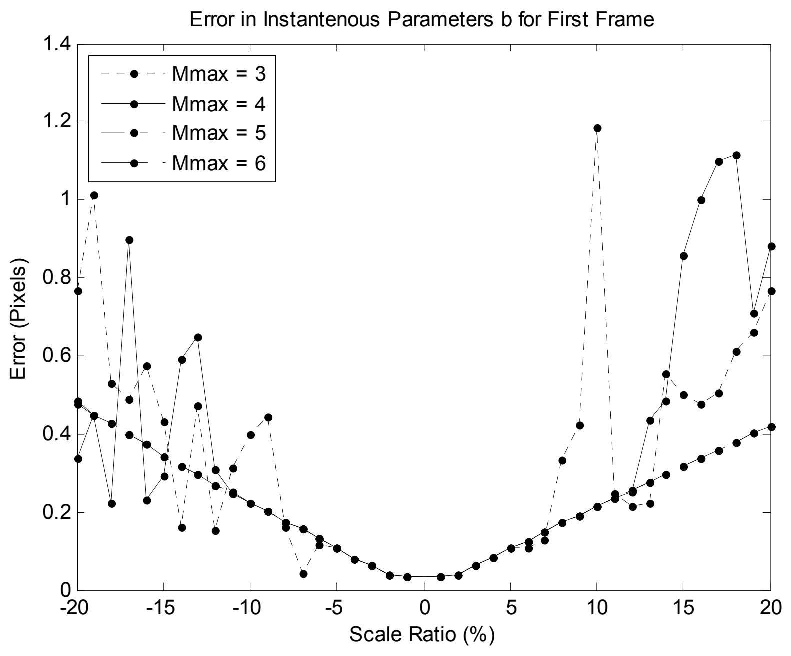


**Figure 6.14** Test 2 – Scale (**b**): Error in the instantaneous parameters $p_5$ and $p_6$ for the Scaling Lenna Test sequence for different number of levels. The errors are computed for scale ratios -20% – +20%.

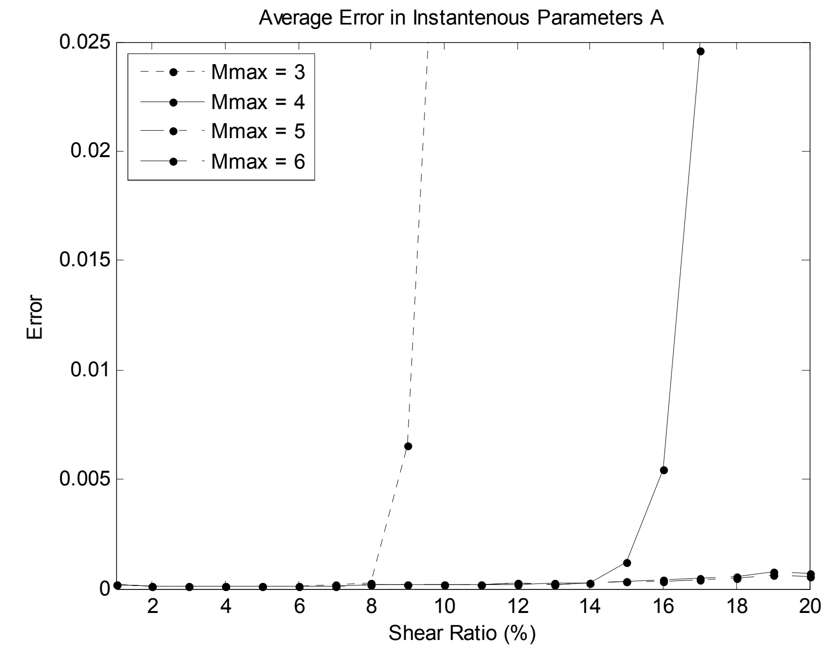


**Figure 6.15** Test 2 – Shear (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Shearing Lenna Test sequence for different number of levels. The errors are computed for shear ratios 1% – 20%. The average is computed over 10 frames.

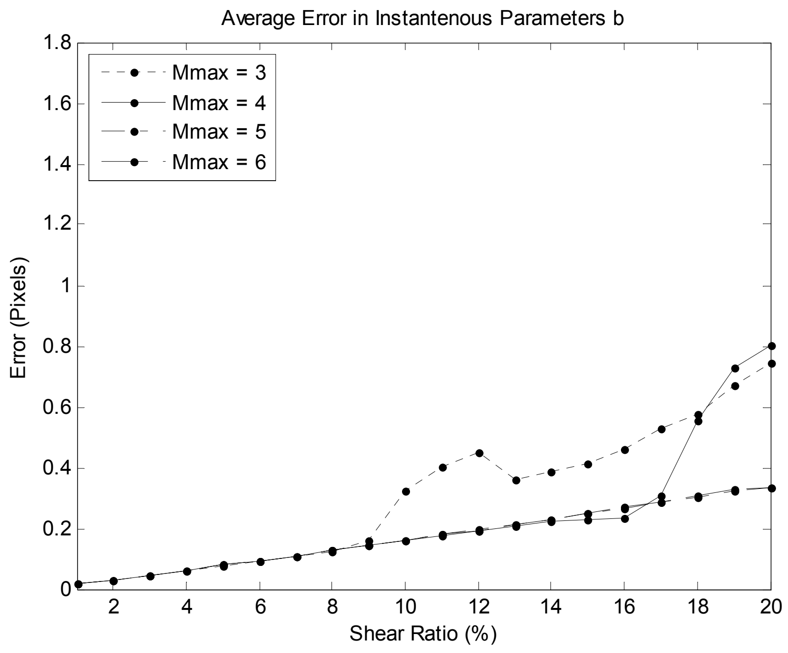


**Figure 6.16** Test 2 – Shear (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Shearing Lenna Test sequence for different number of levels. The errors are computed for shear ratios 1% –20%. The average is computed over 10 frames.

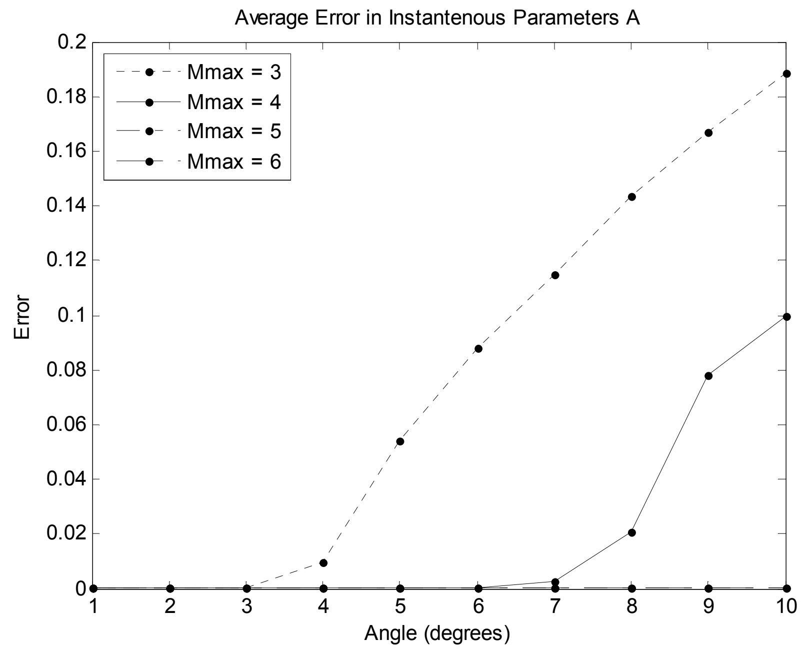


**Figure 6.17** Test 2 – Rotation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Rotating Lenna Test sequence for different number of levels. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

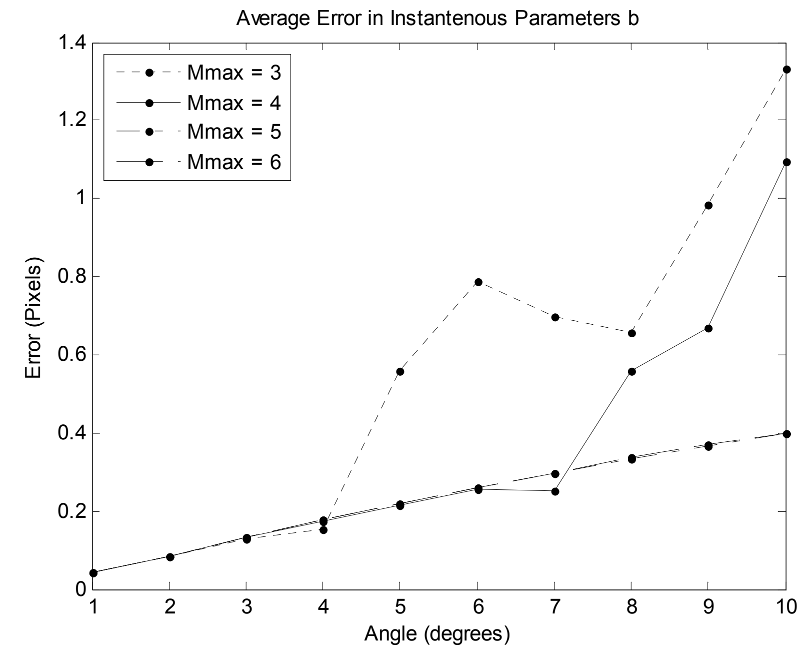


**Figure 6.18** Test 2 – Rotation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Rotating Lenna Test sequence for different number of levels. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

### 6.2.5 Test 3: Comparison of Motion Models

The purpose of these tests is to compare the performance of the motion models with each other and with Lucas-Kanade [57]. The comparison of the different motion models will be in terms of both accuracy and estimation limits; however, the comparision of the models with Lucas-Kanade will be in terms of accuracy only.

The Lenna Test Sequences are used for the tests with a target size of 64 × 64 pixels. For the measurement of errors, the average instantaneous errors for the parameters **A** and **b**, namely (6.14) and (6.15) are used. The results for each motion type are given from Figure 6.20 to Figure 6.27. The relation with the parameters $p_k$ and the affine motion parameters **A** and **b** are given in expressions from (6.5) through (6.7).

For **translation**, similar performance is obtained for all motion types. In terms of accuracy, Lucas-Kanade showed the best performance by taking its advantages of being intensity-based. The Complex Wavelet Tracker, operating in the complex wavelet domain and being approximately shift-invariant, showed a compatible performance to Lucas-Kanade.

For **scale**, the *similarity* motion model performed much better than the *affine* motion model and had similar performance with Lucas-Kanade. This is due to the fact that the *similarity* motion model is simpler than the *affine* motion model, which allows for different scaling of $x$ and $y$ axes and includes shear motion as well, allowing the solution to drift away from pure scaling. For the *translational* motion model, as it does not support scaling, it is expected that it keeps the track gate at the center. From this point of view, the center is maintained within 1 pixel range for scale ratios up to ±10%.

For **shear**, the *affine* motion model performed similar to Lucas-Kanade; whereas the *similarity* motion model had a significant increasing error due to the fact that it does not support shear type of motion. For the *translational* motion model, the target center is maintained within 1 pixel range for shear ratios up to 10%.

In spite of the fact that shear motion is only supported by the *affine* motion model, it is expected from the Complex Wavelet Tracker that the other two models maintain track as much as possible and keep the track gate at the center of the target. For the

*Similarity* motion model, the track gate is rotated and scaled down to match the target as much as possible, which is a notable adaptation of this model. This is depicted in Figure 6.19.

For **rotation**, as with scale, the *similarity* motion model showed a much better performance than the *affine* motion model; and this performance was compatible to Lucas-Kanade. Same reasoning for scale applies to this type as well. For the *translational* motion model, the track gate center is maintained within 1 pixel for only up to 4° of rotation.

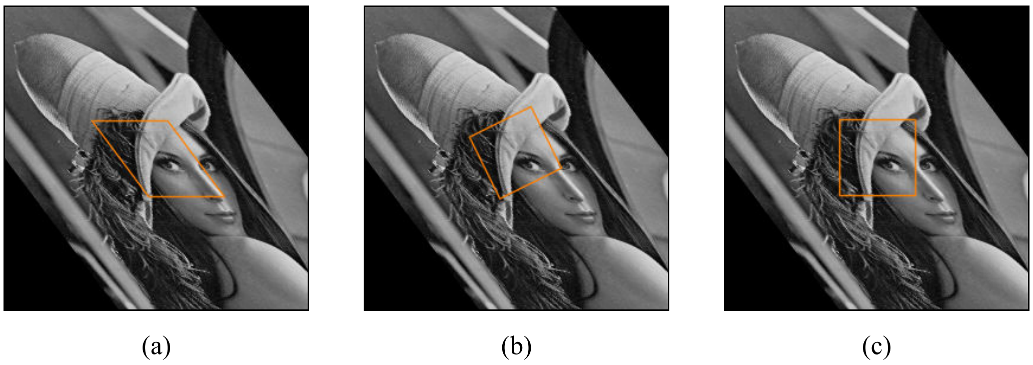


**Figure 6.19** Representative frames from the Shearing Lenna Test Sequence showing the track gate for different motion models. The shear ratio is 5% and the target size is 64 × 64 pixels. The track gate as is initiated at the first frame on the center of the target. The images show the track gate after the 15th frame for (a) the *affine* motion model, (b) the *similarity* motion model, and (c) the *translational* motion model.

On the overall, it is observed that the Complex Wavelet Tracker showed a compatible performance to Lucas-Kanade. The *similarity* motion model performed better for scale and rotation. This can be explained by the fact that the *similarity* motion model is especially designed to reflect scale and rotation and hence the parameters are restricted to a subset of deformations allowed by the *affine* motion model. Therefore, these limitations lead to more accurate estimates by restricting the solution to drift to other types of motion. The *translational* motion model maintained the track gate on the target center within reasonable limits for affine motions.

Generally, for these types of tests where the deformations are perfect where no disturbances are present, the Lucas-Kanade takes advantage of being intensity-based and its iterative nature leads to converge to the correct solution eventually. The Complex Wavelet Tracker, however, operates in the complex wavelet domain, which is approximately shift-invariant and hence approaches to the solution within some limits only.

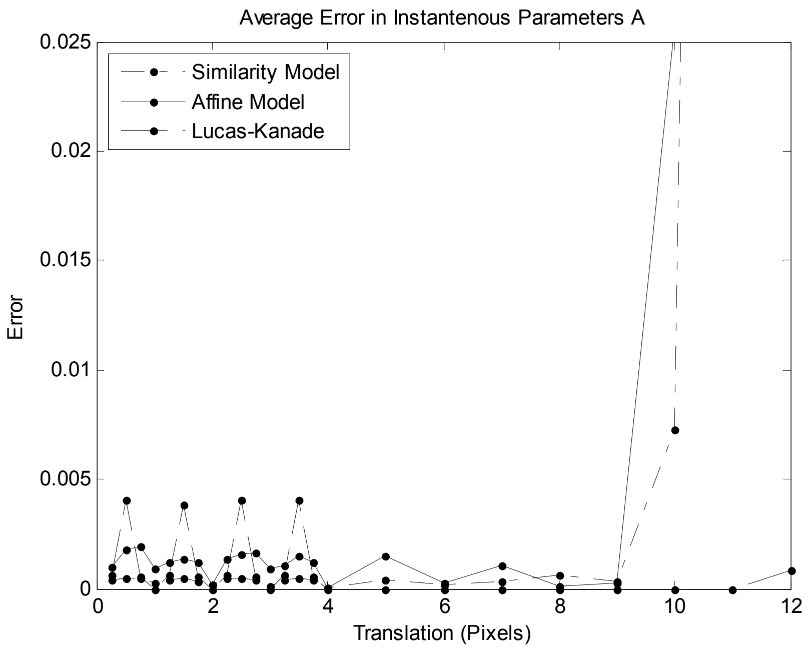


**Figure 6.20** Test 3 – Translation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Translating Lenna Test sequence for different motion models and Lucas-Kanade. Errors are computed for translations between 0.25 – 12 pixels. Average is over 25 frames.

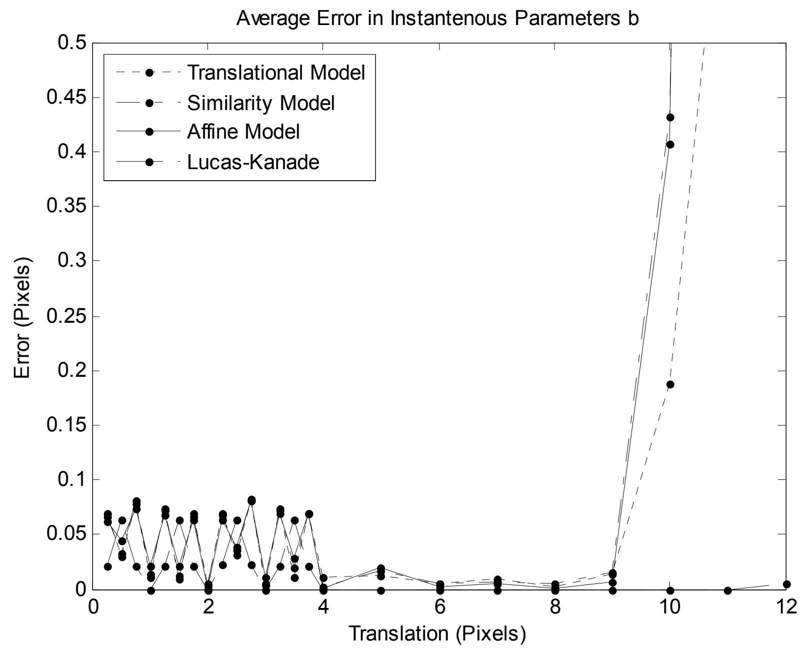


**Figure 6.21** Test 3 – Translation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Translating Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for translations between 0.25 – 12 pixels. The average is over 25 frames.

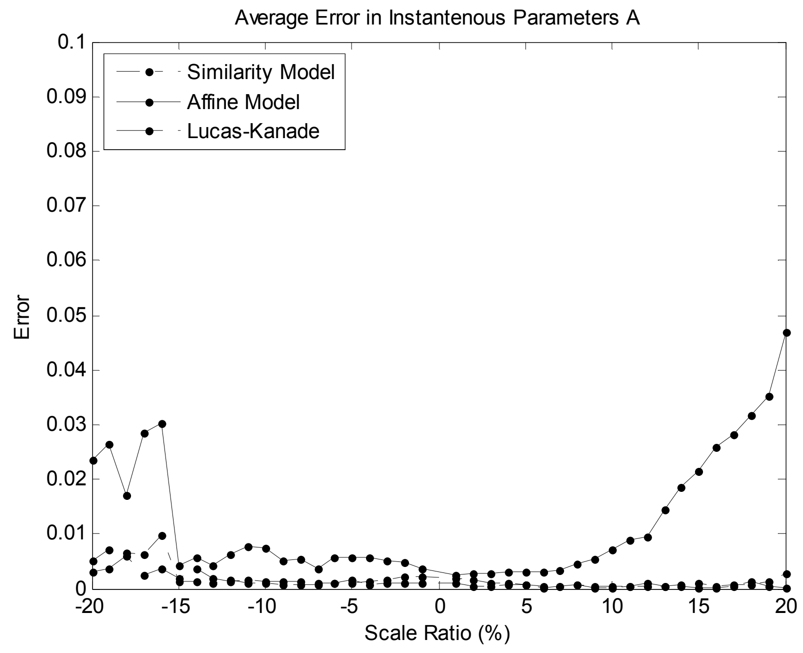


**Figure 6.22** Test 3 – Scale (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Scaling Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

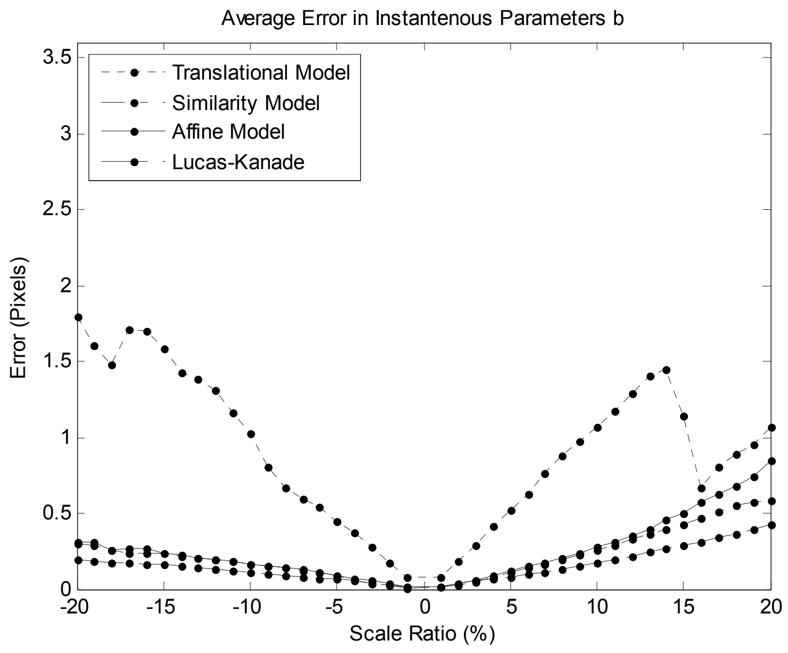


**Figure 6.23** Test 3 – Scale (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Scaling Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

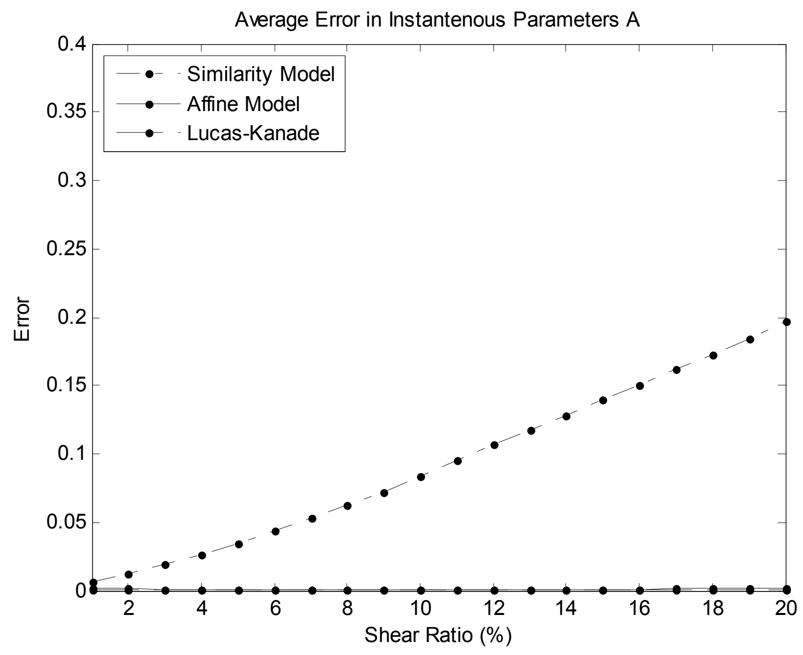


**Figure 6.24** Test 3 – Shear (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Shearing Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for shear ratios 1% – 20%. The average is computed over 10 frames.

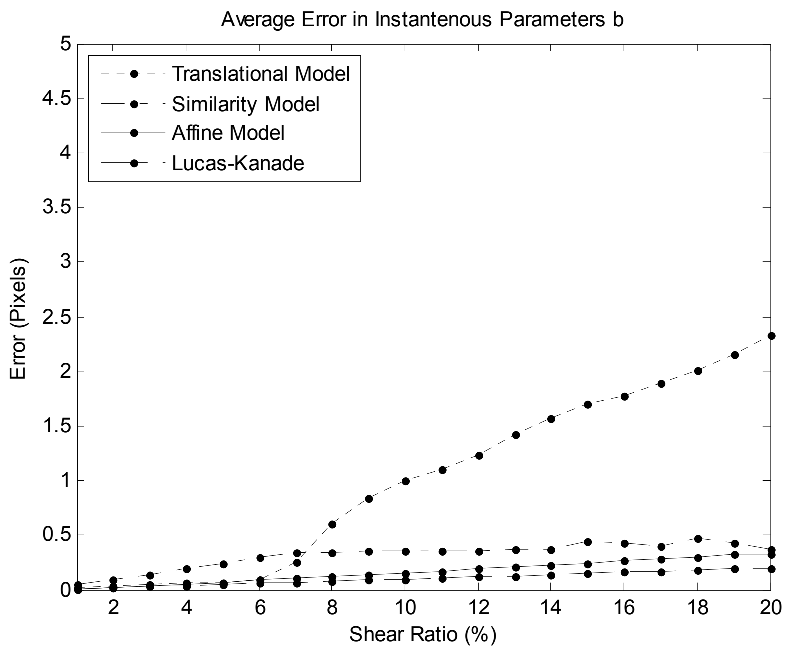


**Figure 6.25** Test 3 – Shear (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Shearing Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for shear ratios 1% –20%. The average is computed over 10 frames.

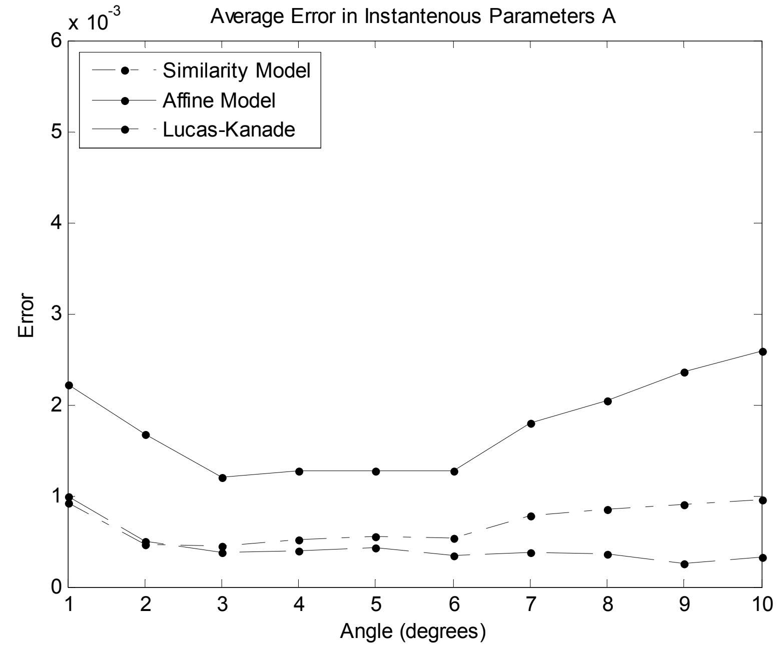


**Figure 6.26** Test 3 – Rotation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Rotating Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

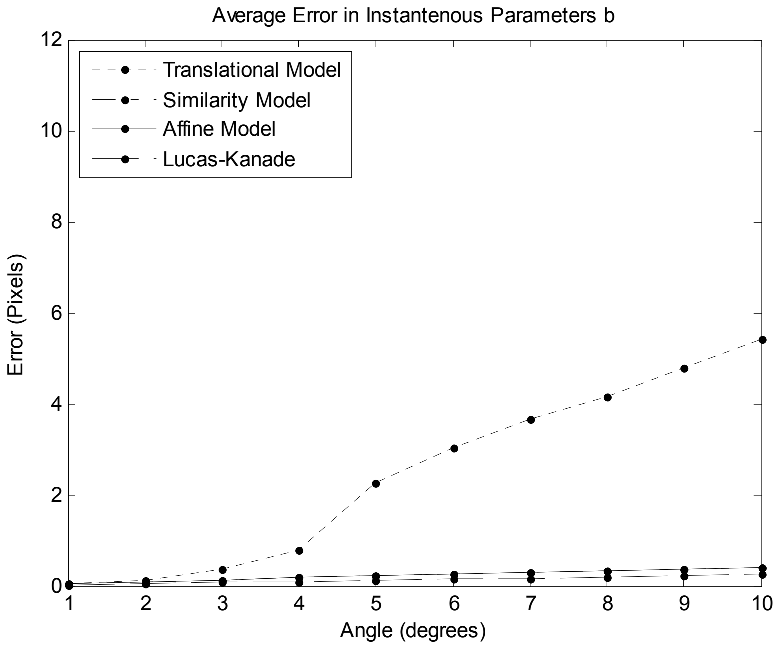


**Figure 6.27** Test 3 – Rotation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Rotating Lenna Test sequence for different motion models and Lucas-Kanade. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

### 6.2.6 Test 4: Robustness to Illumination Changes

The purpose of these tests is to explore the robustness of the proposed algorithm to illumination changes. All three series, the Lenna, Grass, and Cloud Test Sequences are used for the tests. The Lucas-Kanade [57] algorithm is used to compare the results.

The *affine* motion model with 64 × 64 pixels target size is used to perform these tests. The tests are all performed for the four types of motion. For the evaluations, the instantaneous error measures, given in Equations (6.12) and (6.13), are used.

In order to simplify the graphics, we picked one representative case for each motion type: 4 pixels for translation, 5% for scale, 5% for shear, and 4° for rotation. The errors of each motion type are then averaged to obtain a single error value for each disturbance value. The results are given in the next two subsections for each series separately with comparison to the Lucas-Kanade algorithm.

##### 6.2.6.1.1 Brightness: Additive Offset

An offset ranging from 0 to 25 is added to the sequences at each frame. The results are shown in Figure 6.28 through Figure 6.33 for Lenna, Grass, and Cloud series.

It is observed that for the Complex Wavelet Tracker, the accuracy in the track parameters are not affected for all the three series. Being a phase-based approach, this type of disturbance to some extend does not affect the performance of the Complex Wavelet Tracker, as expected. However, for the Lucas-Kanade algorithm, although more precise at very low brightness disturbance levels, the performance gradually decreases as the additive offset increases. In the Cloud series, the track is lost above an offset value around 10 for Lucas-Kanade. This behaviour is normal for intensity-based methds, as the intensity is being changed in these types of disturbances.

##### 6.2.6.1.2 Contrast: Intensity Scaling

The intensity of the sequences is scaled by a factor ranging from 0.75 to 1.25. The results are shown in Figure 6.34 through Figure 6.39 for the Lenna, Grass, and Cloud series.

It is observed that for the Complex Wavelet Tracker, the accuracy in the track parameters are not affected for all the three series. However, for the Lucas-Kanade algorithm, although more precise at low scaling factors, the performance gradually decreases as the scaling factor increases. In the Cloud series, the track is lost above an intensity factor around ±0.10 for Lucas-Kanade. This shows again that phase-based approaches are robust to intensity-perturbations in contrast to intensty-based approaches, as expected.

#### 6.2.6.1.3 Summary of the Results

Simulations have been performed in order to explore the affects of brightness and contrast changes on the performance of the Complex Wavelet Tracker. The results are compared with the Lucas-Kanade [57] algorithm.

On the overall, it is observed that for both brightness and contrast changes the proposed algorithm showed excellent robustness by showing no degradation in the accuracy of its estimations to a large extend. In comparison, the performance of the Lucas-Kanade method, however, showed a gradual decrease in the precision of its estimations with increasing perturbations. For the Cloud sequence, it showed loss of track after some degree of disturbance. Since the Cloud series, in contrast to the other two series, is a very smooth image with no edges or corners, is not suitable for Lucas-Kanade in the first case. And, the addition of intensity perturbations creates these losses.

This outcome is somehow expected. The Complex Wavelet Tracker operates in the complex wavelet domain and bases its estimations on the phase information. The phase information is known to be immune to illumination changes [30], which makes the Complex Wavelet Tracker robust to illumination. The Lucas-Kanade [57] algorithm, however, uses the intensity information to perform its estimations and hence, is easily affected by illumination changes.

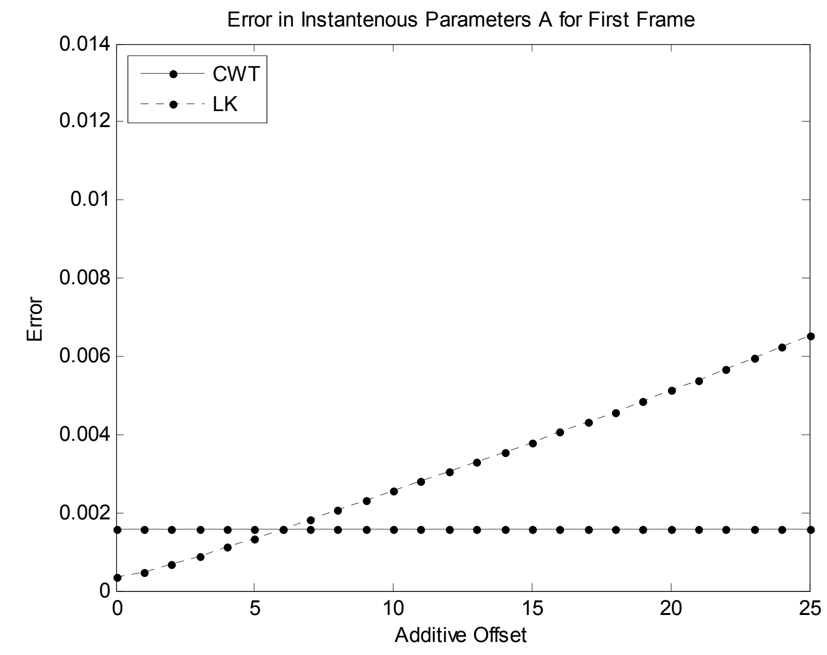


**Figure 6.28** Test 4 Lenna – Brightness (**A**): Robustness to brightness changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

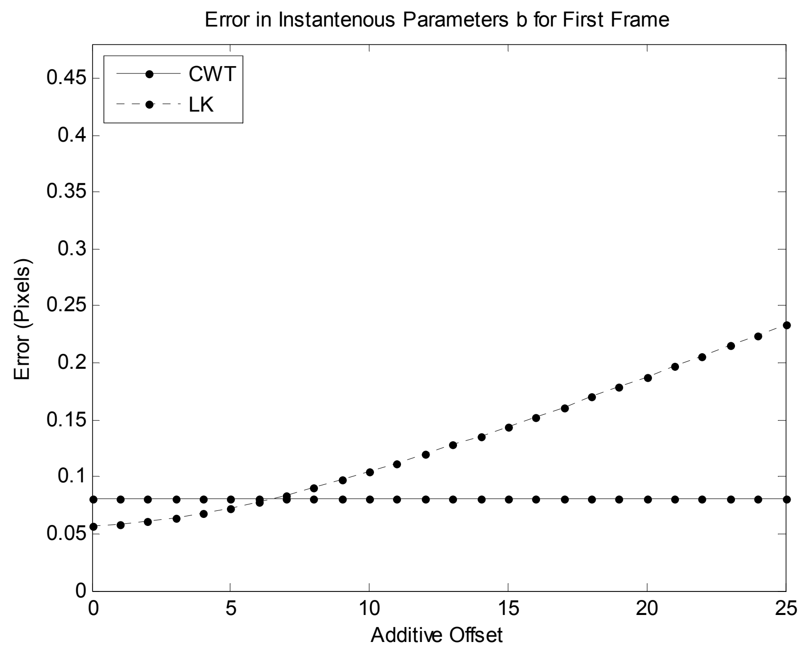


**Figure 6.29** Test 4 Lenna – Brightness (**b**): Robustness to brightness changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

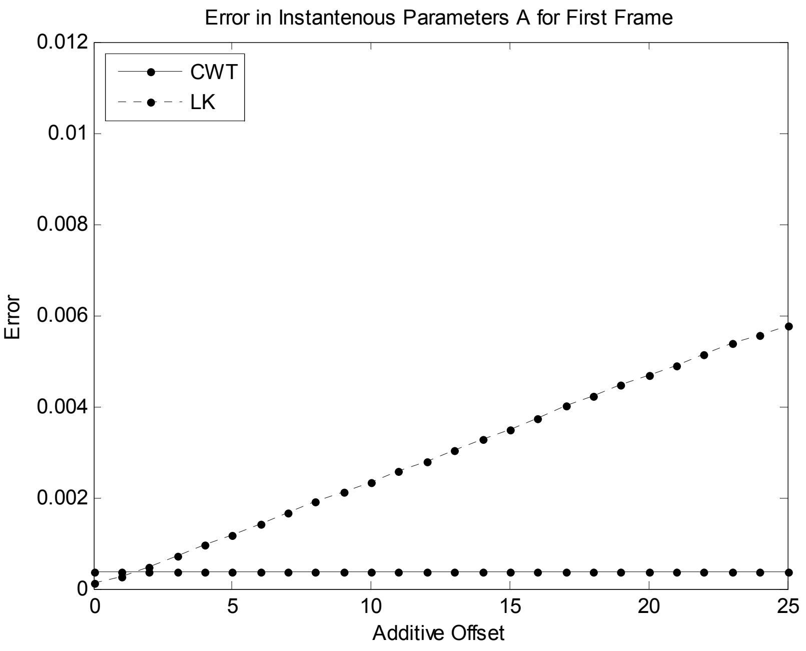


**Figure 6.30** Test 4 Grass – Brightness (**A**): Robustness to brightness changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

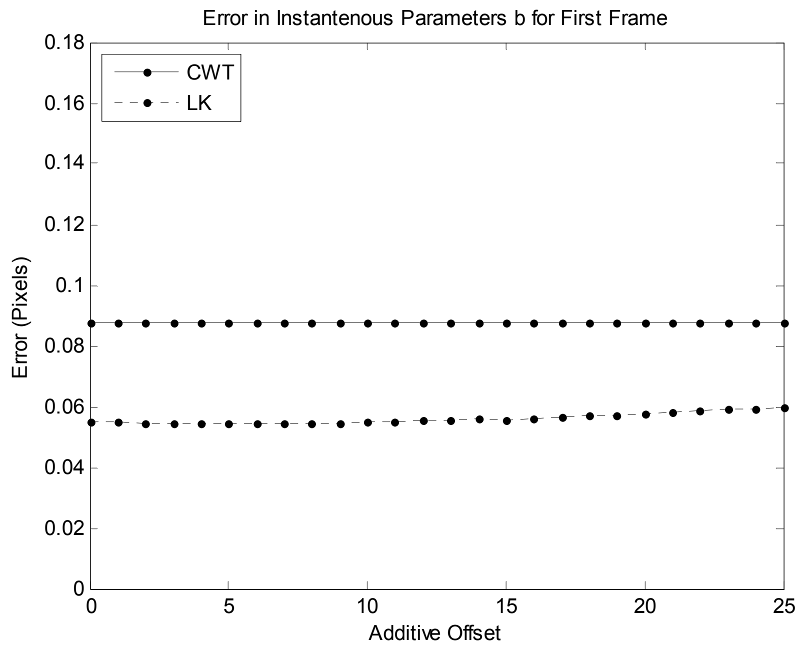


**Figure 6.31** Test 4 Grass – Brightness (**b**): Robustness to brightness changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

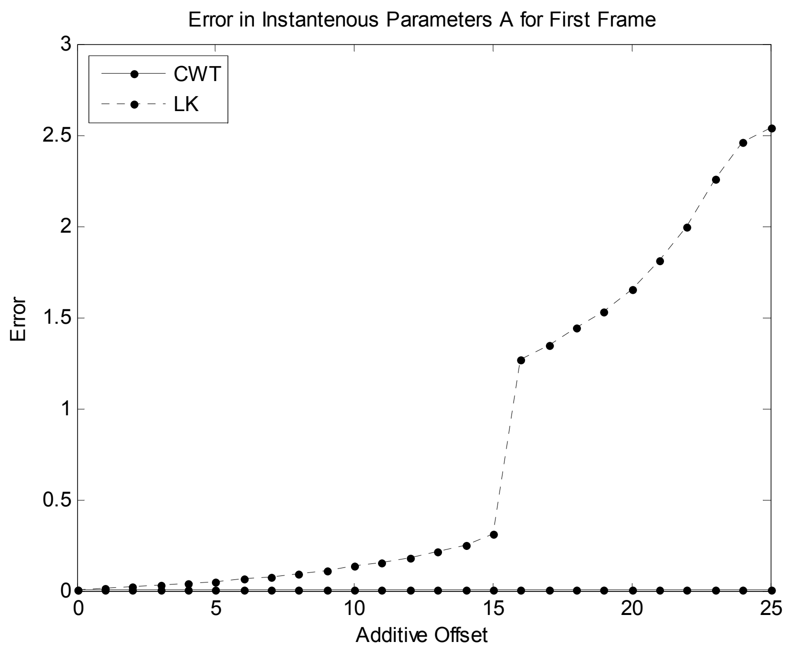


**Figure 6.32** Test 4 Cloud – Brightness (**A**): Robustness to brightness changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

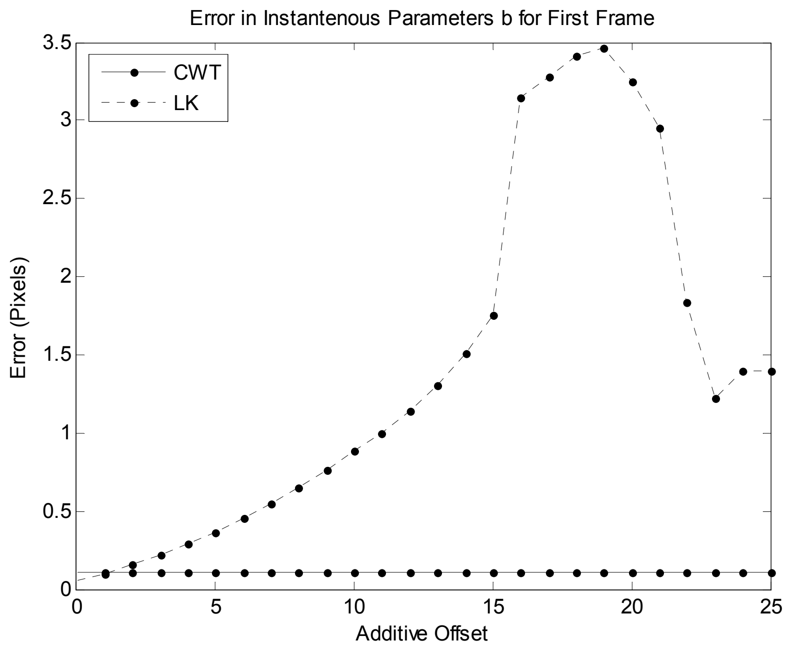


**Figure 6.33** Test 4 Cloud – Brightness (**b**): Robustness to brightness changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

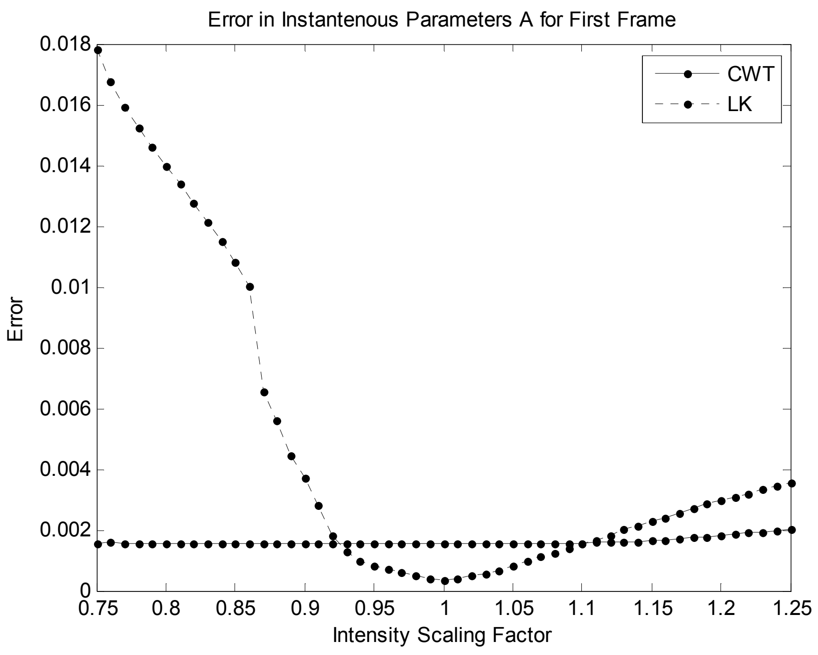


**Figure 6.34** Test 4 Lenna – Contrast (**A**): Robustness to contrast changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

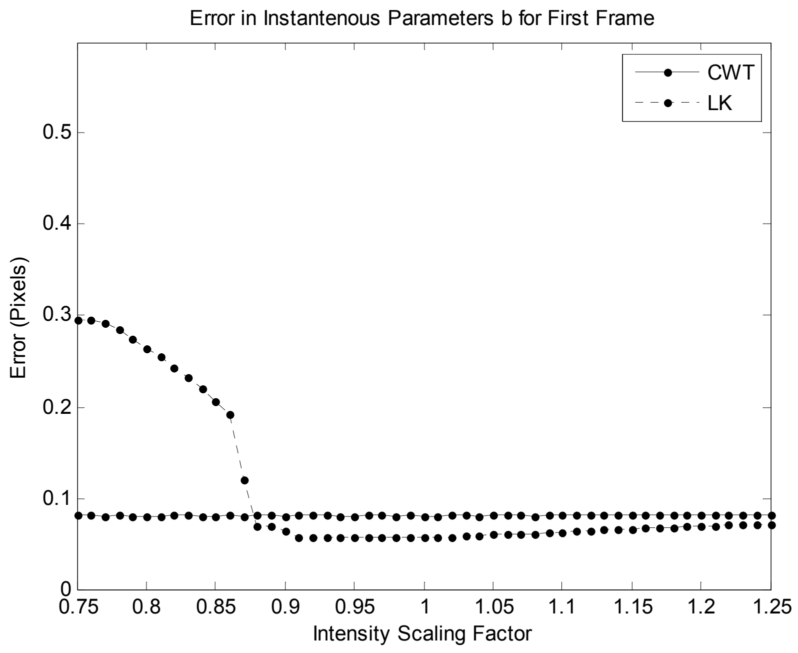


**Figure 6.35** Test 4 Lenna – Contrast (**b**): Robustness to contrast changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

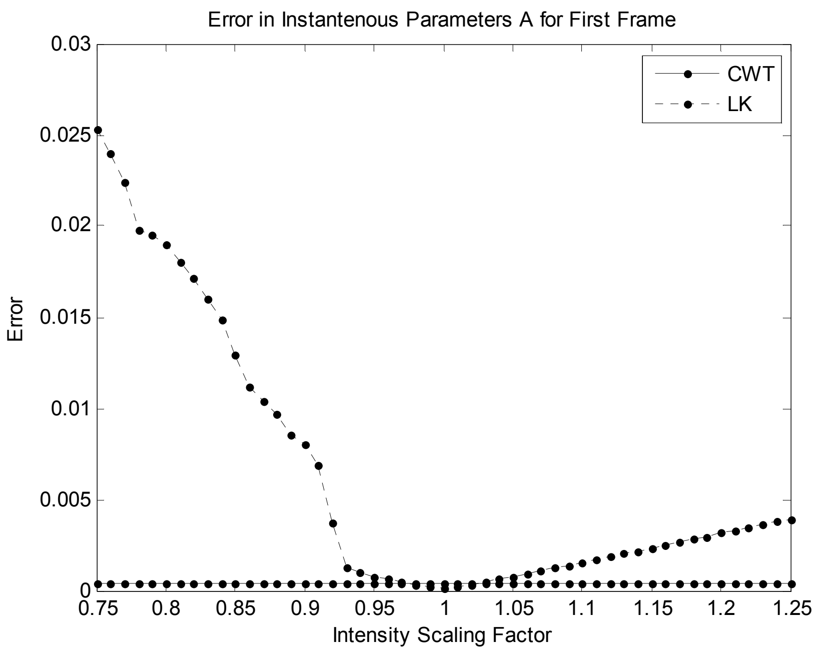


**Figure 6.36** Test 4 Grass – Contrast (**A**): Robustness to contrast changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

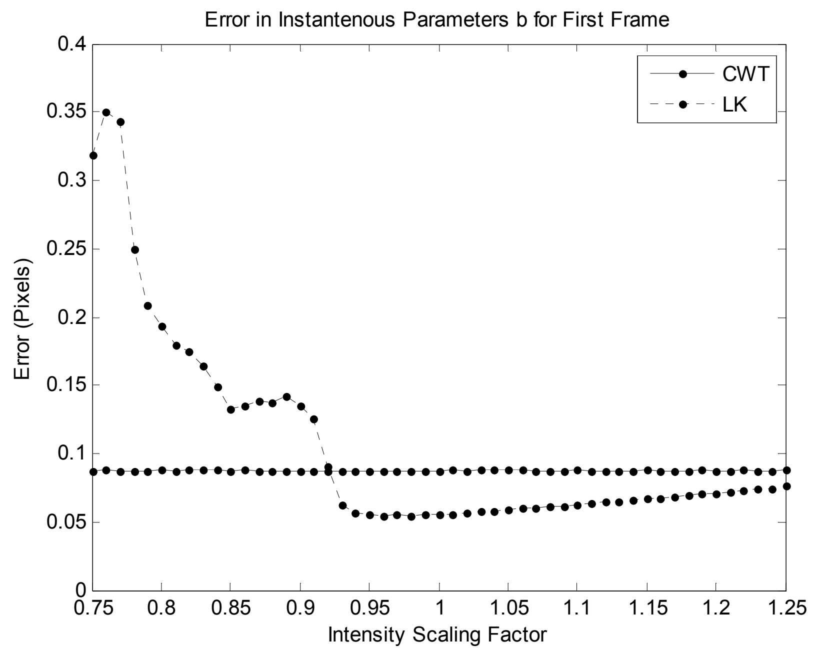


**Figure 6.37** Test 4 Grass – Contrast (**b**): Robustness to contrast changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

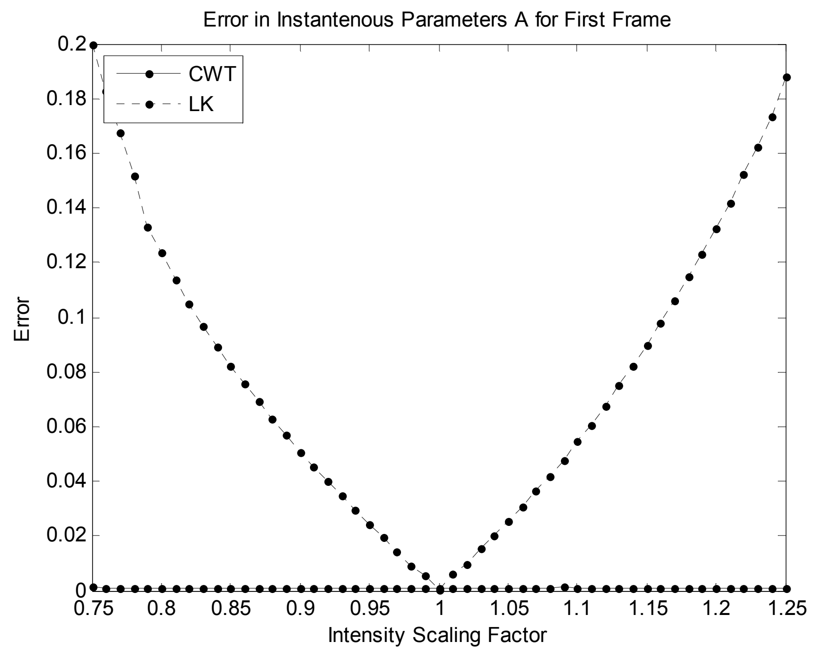


**Figure 6.38** Test 4 Cloud – Contrast (**A**): Robustness to contrast changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

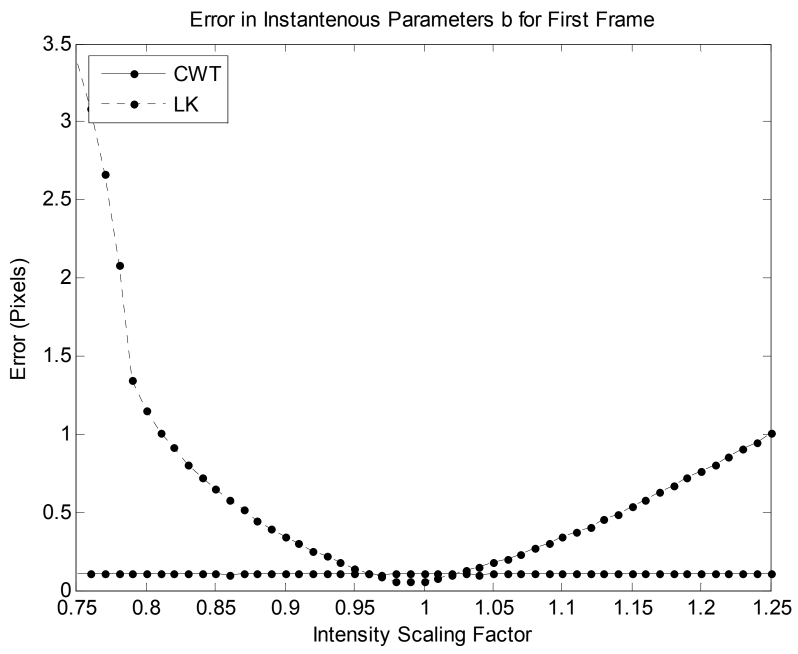


**Figure 6.39** Test 4 Cloud – Contrast (**b**): Robustness to contrast changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

### 6.2.7 Test 5: Robustness to Noise

The purpose of these tests is to explore the robustness of the proposed algorithm to additive white Gaussian noise. All three series, the Lenna, Grass, and Cloud Test Sequences are used for the tests. The Lucas-Kanade [57] algorithm is used to compare the results.

Additive zero-mean white Gaussian noise with standard deviation varying from 0 to 20 is added to the sequences. The *affine* motion model with 64 × 64 pixels target size is used to perform these tests. The tests are all performed for the four types of motion. For the evaluations, the average instantaneous error measures, given in Equations (6.14) and (6.15), are used.

In order to simplify the graphics, we picked one special example for each motion type: 4 pixels for translation, 5% for scale, 5% for shear, and 4° for rotation. The errors of each motion type are then averaged to obtain a single error value for each noise level. The results are given for each series separately with comparison to the Lucas-Kanade algorithm in Figure 6.40 through Figure 6.45.

It is observed that for all the motion types and for all test series, the error in the track parameters increases gradually as the noise increases. This behavior is similar for the Lucas-Kanade algorithm. The overall precision of Lucas-Kanade is better. With increasing noise; this situation does not change for the Lenna series, where Lucas-Kanade performs quite robustly. However, for the Grass and Cloud series, the error in the affine parameter **A** is similar for both algorithms, where for low noise levels Lucas-Kanade, and for high noise levels the Complex Wavelet Tracker peforms slightly better.

As a summary, the robustness of the proposed method to additive noise is similar to Lucas-Kanade [57]. The estimation error increases as the noise standard deviation increases. The Complex Wavelet Tracker shows more immunity to noise for smooth intensity regions as in the Cloud series, in contrast to Lucas-Kanade.

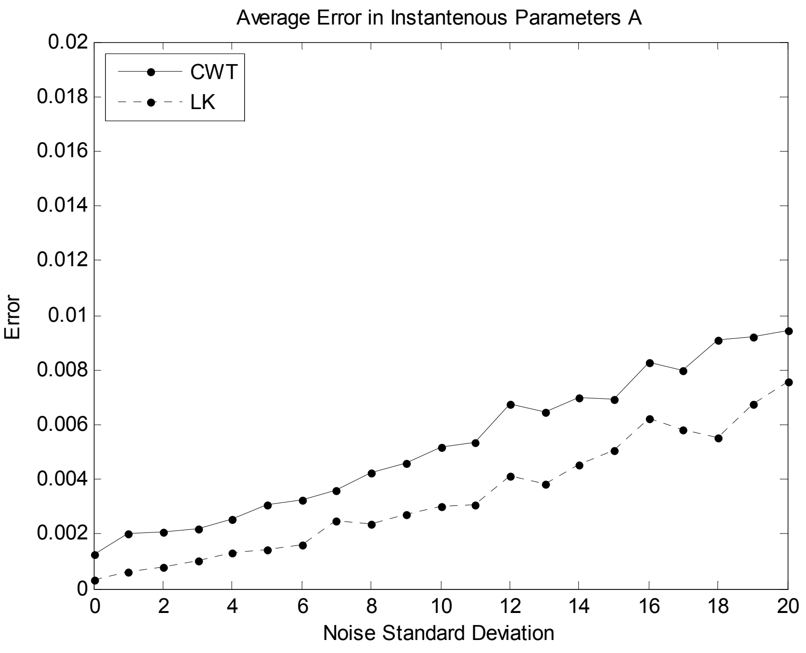


**Figure 6.40** Test 5 Lenna – Noise (**A**): Robustness to noise. The average error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

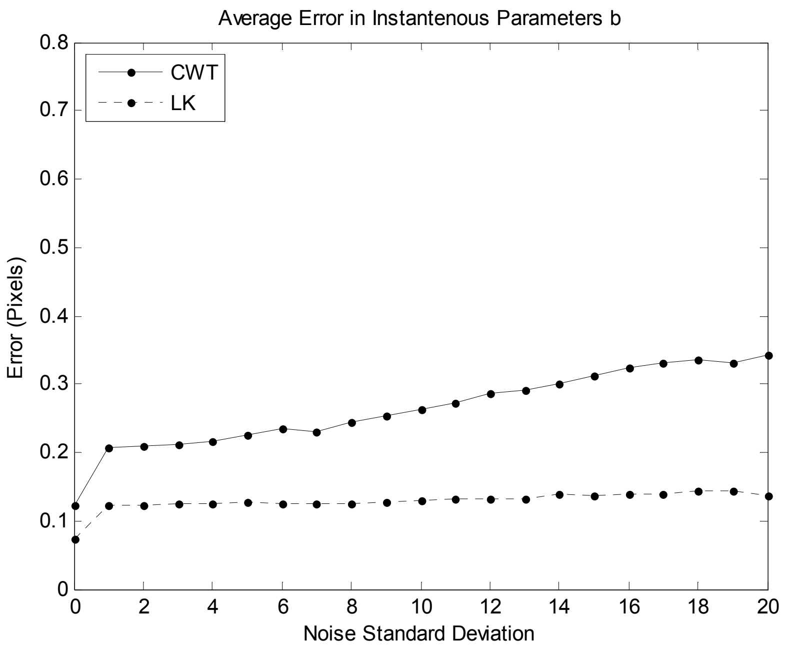


**Figure 6.41** Test 5 Lenna – Noise (**b**): Robustness to noise. The average error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

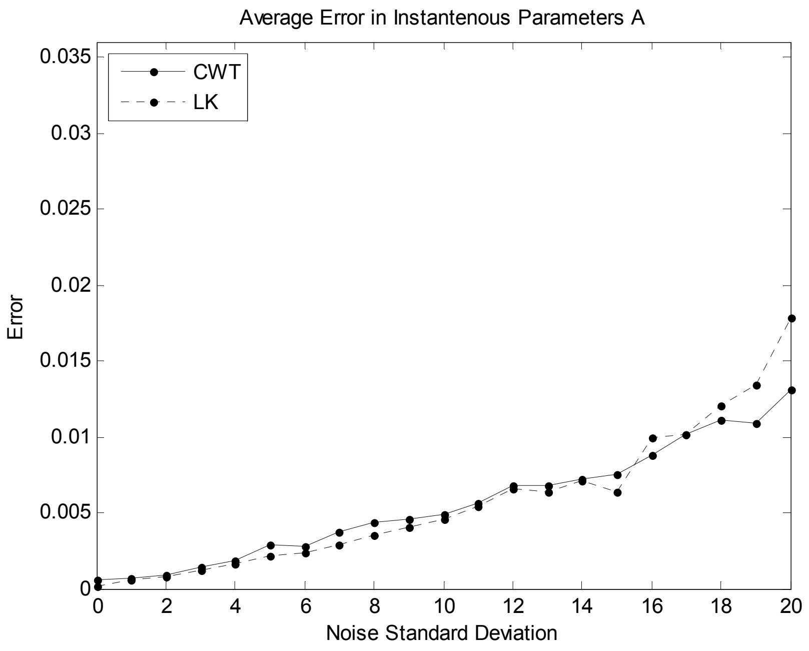


**Figure 6.42** Test 5 Grass – Noise (**A**): Robustness to noise. The average error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

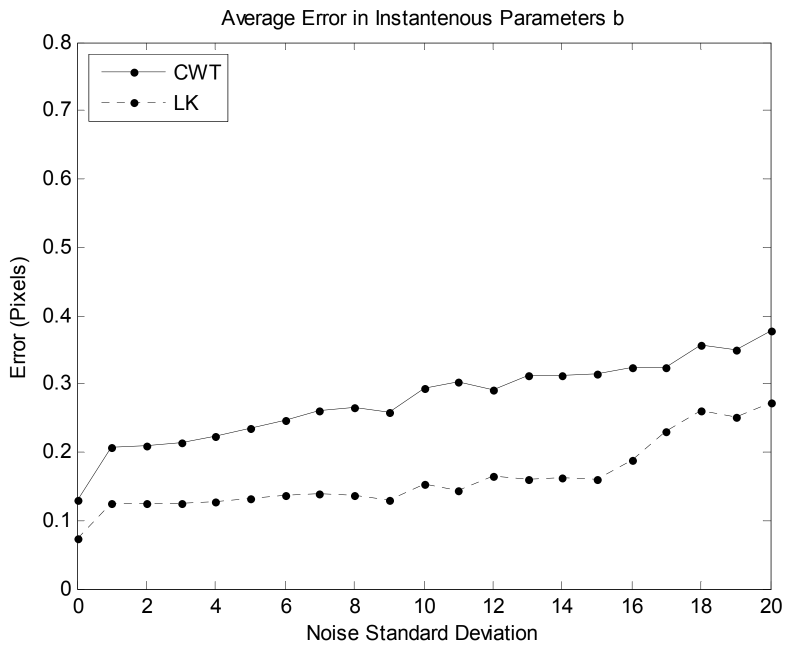


**Figure 6.43** Test 5 Grass – Noise (**b**): Robustness to noise. The average error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

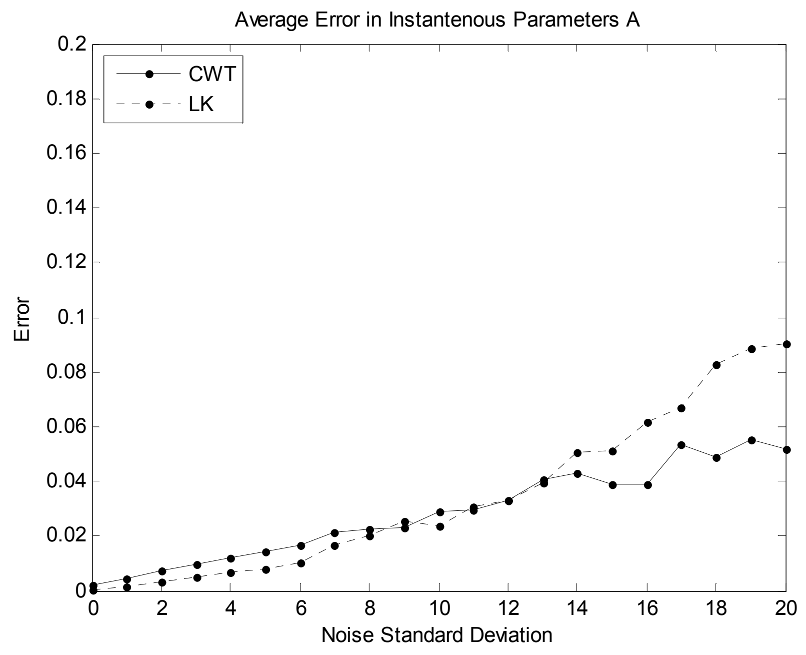


**Figure 6.44** Test 5 Cloud – Noise (**A**): Robustness to noise. The average error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

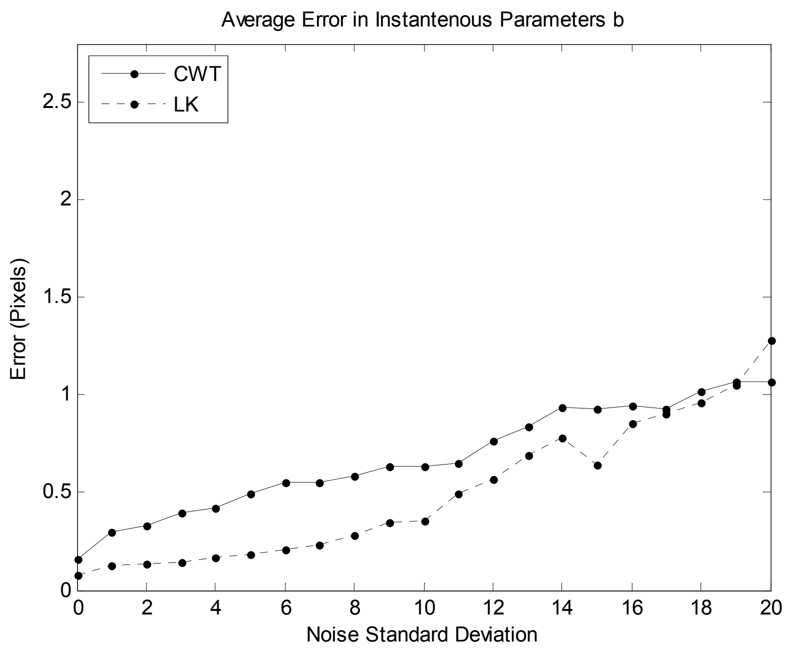


**Figure 6.45** Test 5 Cloud – Noise (**b**): Robustness to noise. The average error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

### 6.2.8 Test 6: Robustness to Intensity Patterns

The purpose of these tests is to explore the robustness of the accuracy of the Complex Wavelet Tracker under different intensity patterns, or textures. The Lenna, Grass, and Cloud Test Sequences are used for the tests to reflect different intensity patterns. The results are compared with the results of the Lucas-Kanade [57] algorithm.

The *affine* motion model with 64 × 64 pixels target size is used to perform these tests. Three different types of test are performed. First, the effect of intensity patterns on the affine estimation limits is examined for each motion type. Second and third, tests are performed to explore the effect on the robustness of the algorithm to noise and illumination changes. The results for each test are given in the following subsections.

#### 6.2.8.1 Effect of Intensity Patterns to Affine Limits

The purpose of these tests is to evaluate the change in the accuracy and affine estimation limits of the algorithm under different intensity patterns, or textures.

For the measurement of errors, the average instantaneous errors for the parameters **A** and **b**, namely (6.14) and (6.15), are used. The *affine* motion model is used to perform these tests. A comparison to Lucas-Kanade [57] method is also provided. Simulations have been performed on the Lenna, Grass, and Cloud series using a 64 × 64 pixel-size target.

Results are presented from Figure 6.46 to Figure 6.61 for each motion type one by one. First, the results for the Complex Wavelet Tracker are given, which are followed by the results for Lucas-Kanade. The results are presented for the parameters **A** and **b** separately.

For **translation**, the performance of the Complex Wavelet Tracker for all the series is similar. For the Lucas-Kanade, a break for the Grass series occurs earlier (after 7 pixels) than the other two. This break can be caused due to the repetitive texture of the Grass Series. Besides this break, the performance is close over the full test range.

For **scale**, the Complex Wavelet Tracker behaves similarly from –20% to +10%, but after +10%, the algorithm breaks up for Lenna first, Cloud second, and Grass at last.

This behavior shows that repetitive texture as in the Grass series favors for estimating scale type of deformations. For Lucas-Kanade, however, the performance is similar for all three series.

For **shear**, both algorithms behave similarly for all series. The Complex Wavelet Tracker experiences difficulty for the Cloud series after around 15%. The proposed method performs slightly more accurate for the Grass series than the Lenna series.

For **rotation**, both algorithms present similar results among the three series. For the Complex Wavelet Tracker, the accuracy is much better for the Grass series than the other two series, as in the scale case.

On the overall, it is observed that the affine motion estimation performance of the Complex Wavelet Tracker is similar for different types of textures like the Lenna, Grass, and the Cloud series. There is a small difference in the errors for scale and shear; and for rotation, the error in the Grass series is quite smaller than the others. For the Lucas-Kanade, similar differences are also observed for scale, shear, and rotation. Early breaks occur for Lenna in scale, and for Cloud in shear for the Complex Wavelet Tracker; and for Grass in translation for Lucas-Kanade. Differences in the accuracies show that phase-based methods perform little more confident in repetitive type textures than intensity-based methods. As a result, both methods show compatible performance in terms of robustness to different intensity patterns.

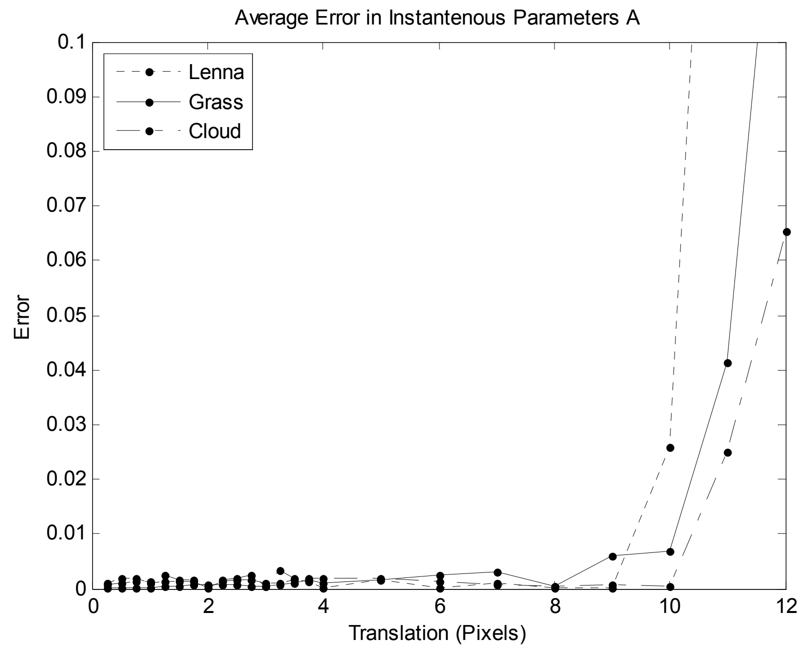


**Figure 6.46** Test 6 CWT – Translation (**A**): Average error in the instantaneous parameters $p_1$ – $p_4$ for the Translating Test sequence for Lenna, Grass, and Cloud. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

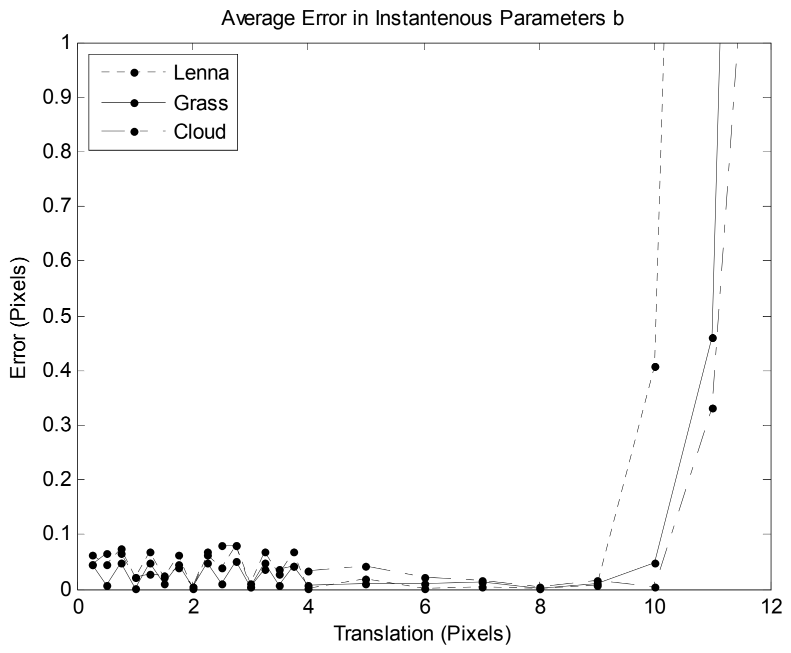


**Figure 6.47** Test 6 CWT – Translation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Translating Test sequence for Lenna, Grass, and Cloud. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

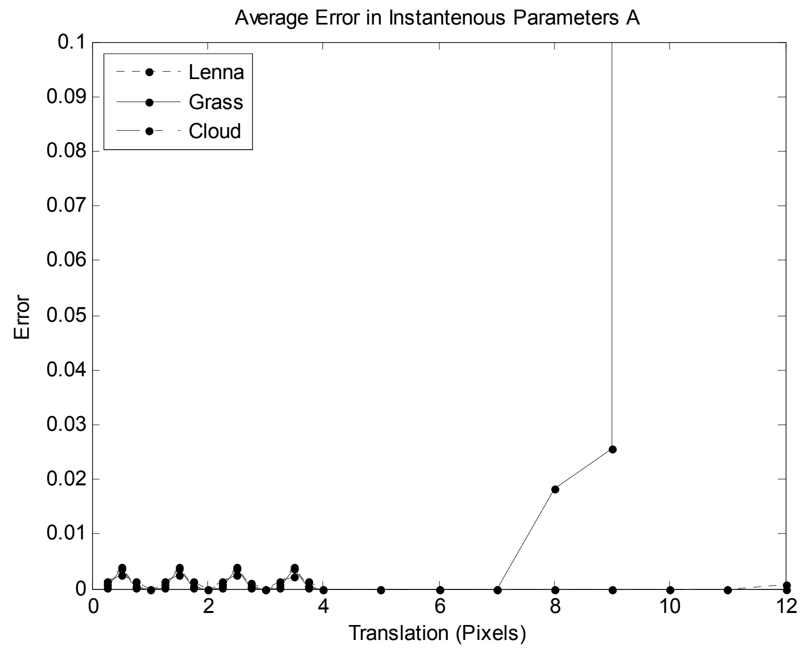


**Figure 6.48** Test 6 LK – Translation (**A**): Average error in the instantaneous parameters $p_1$ – $p_4$ for the Translating Test sequence for Lenna, Grass, and Cloud. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

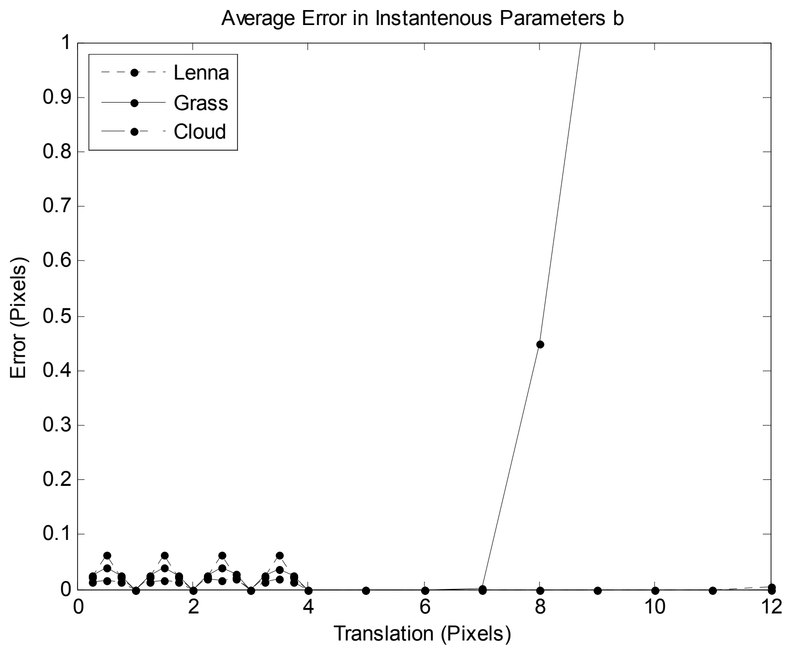


**Figure 6.49** Test 6 LK – Translation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Translating Test sequence for Lenna, Grass, and Cloud. The errors are computed for translations between 0.25 – 12 pixels. The average is computed over 25 frames.

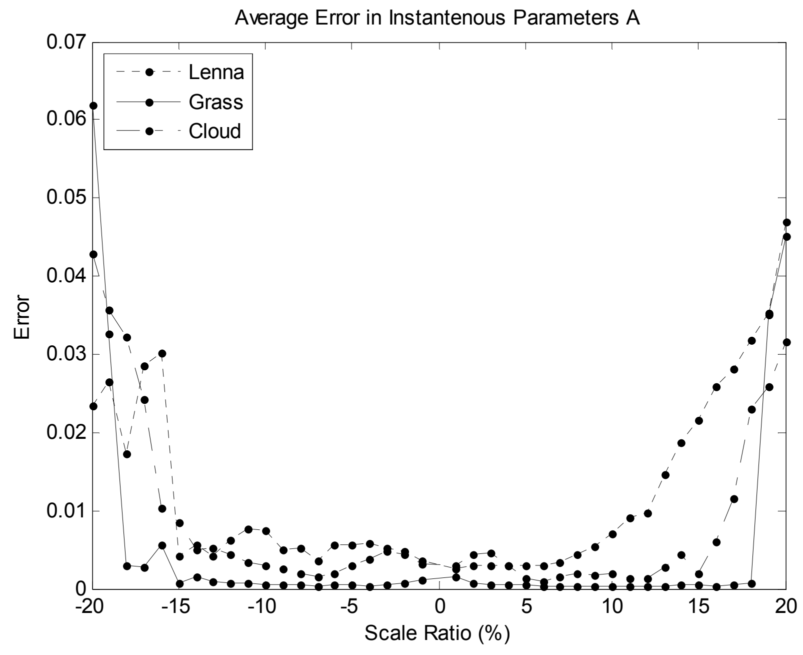


**Figure 6.50** Test 6 CWT – Scale (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Scaling Test sequence for Lenna, Grass, and Cloud. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

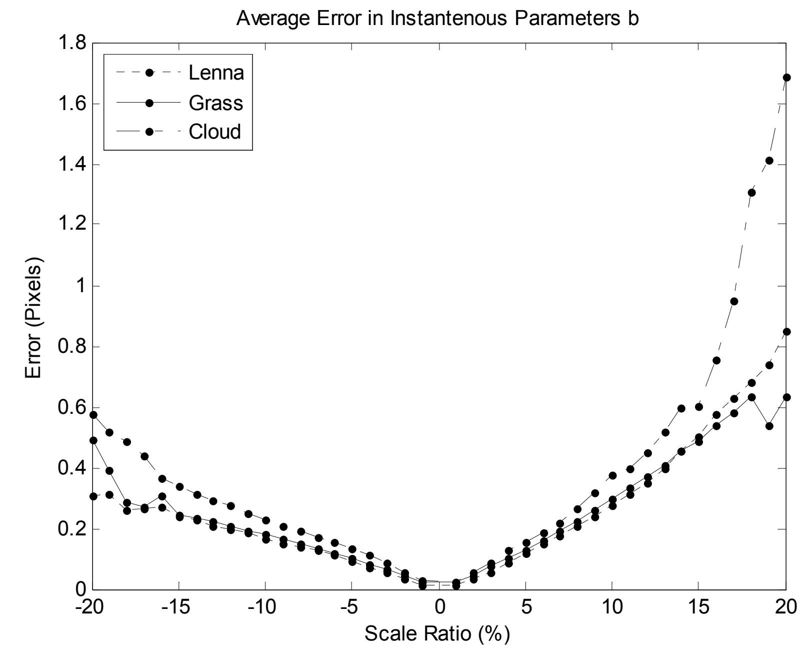


**Figure 6.51** Test 6 CWT – Scale (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Scaling Test sequence for Lenna, Grass, and Cloud. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

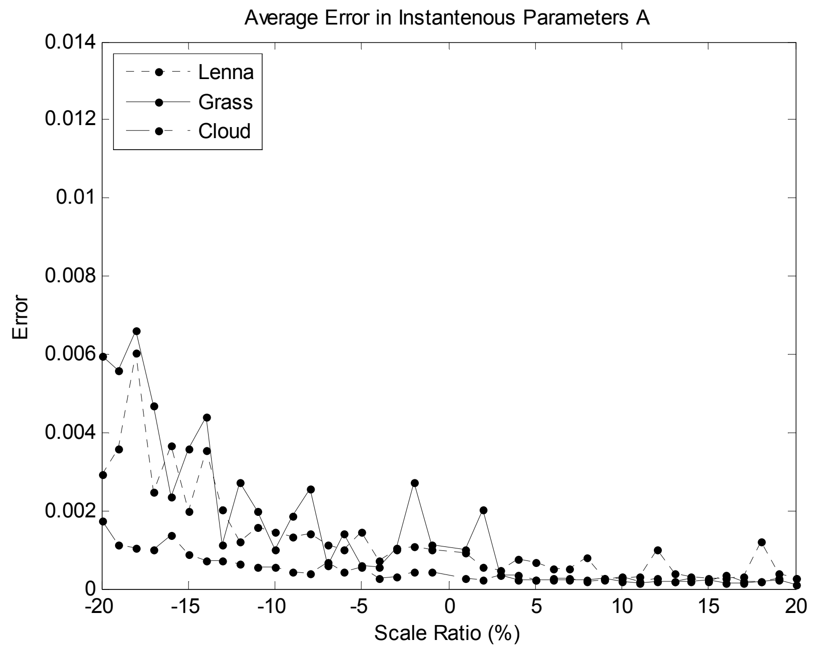


**Figure 6.52** Test 6 LK – Scale (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Scaling Test sequence for Lenna, Grass, and Cloud. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

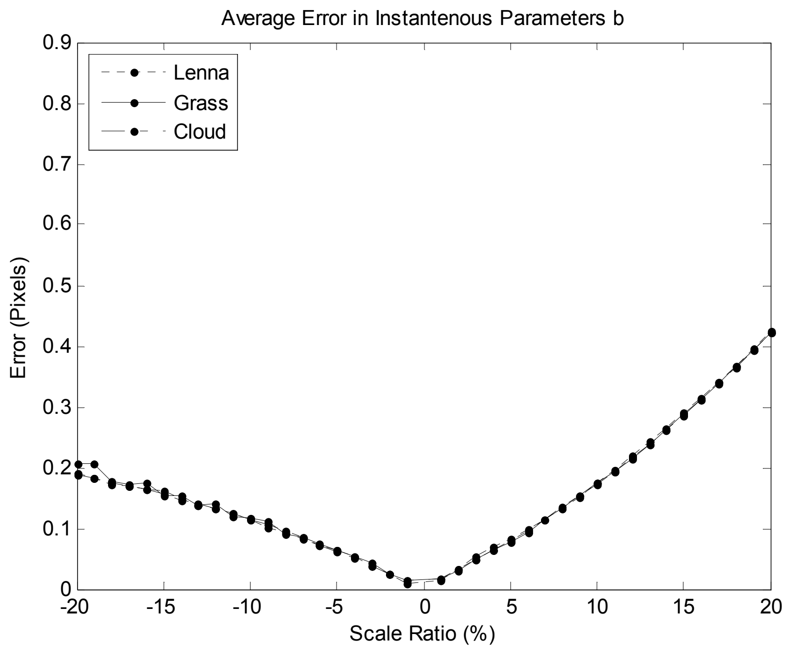


**Figure 6.53** Test 6 LK – Scale (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Scaling Test sequence for Lenna, Grass, and Cloud. The errors are computed for scale ratios -20% – +20%. The average is computed over 5 frames.

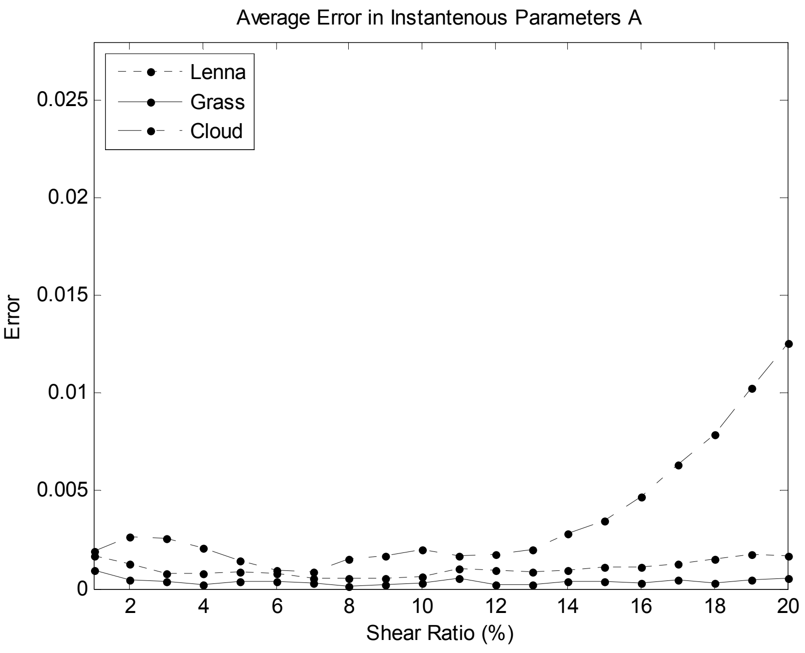


**Figure 6.54** Test 6 CWT – Shear (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Shearing Test sequence for Lenna, Grass, and Cloud. The errors are computed for shear ratios 1% – 20%. The average is computed over 10 frames.

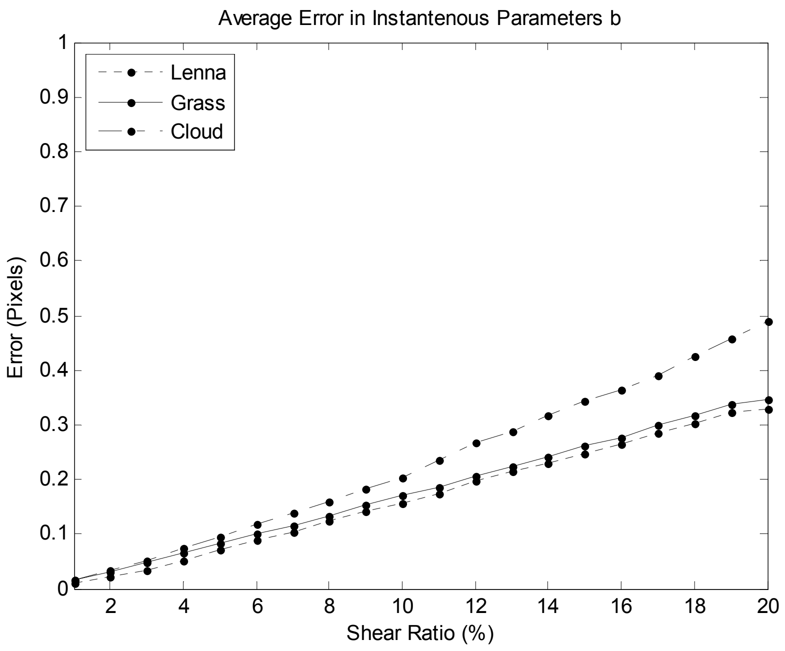


**Figure 6.55** Test 6 CWT – Shear (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Shearing Test sequence for Lenna, Grass, and Cloud. The errors are computed for shear ratios 1% –20%. The average is computed over 10 frames.

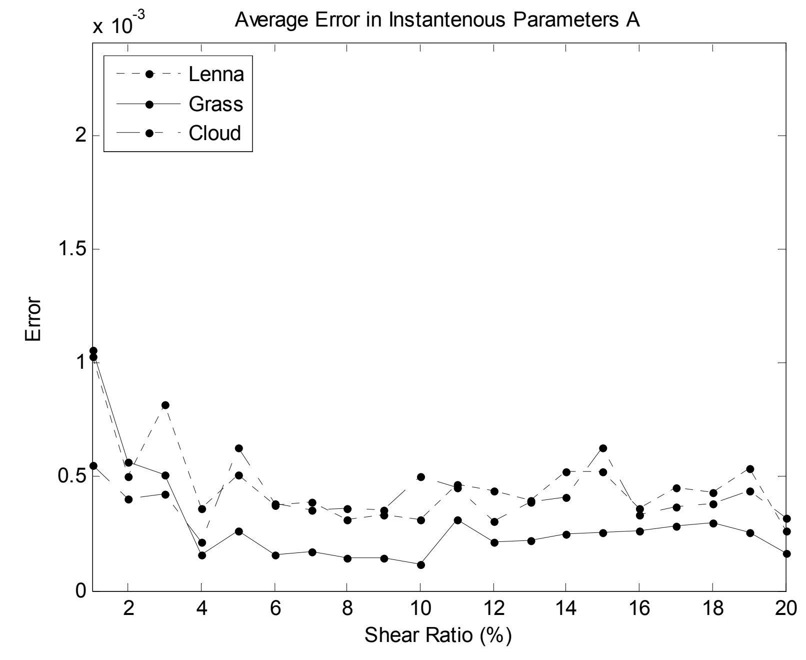


**Figure 6.56** Test 6 LK – Shear (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Shearing Test sequence for Lenna, Grass, and Cloud. The errors are computed for shear ratios 1% – 20%. The average is computed over 10 frames.

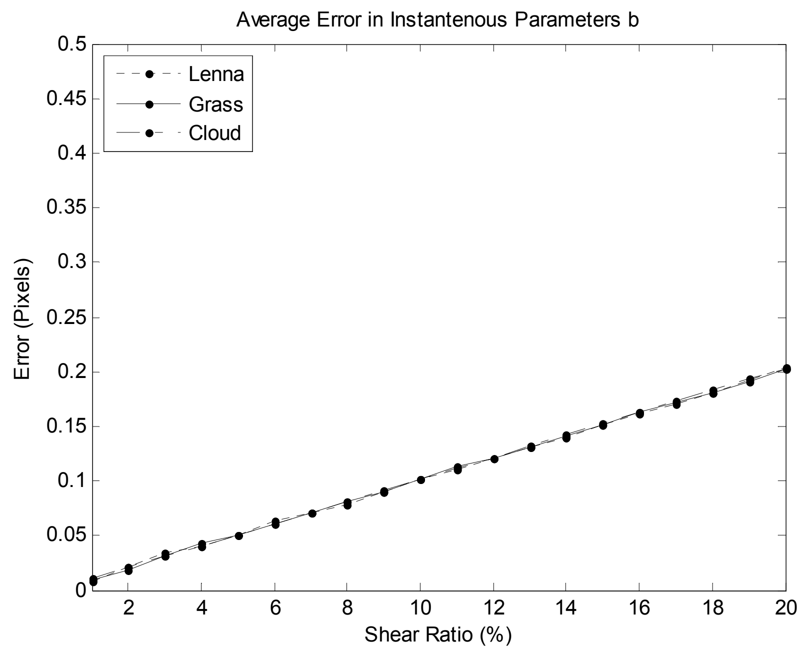


**Figure 6.57** Test 6 LK – Shear (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Shearing Test sequence for Lenna, Grass, and Cloud. The errors are computed for shear ratios 1% –20%. The average is computed over 10 frames.

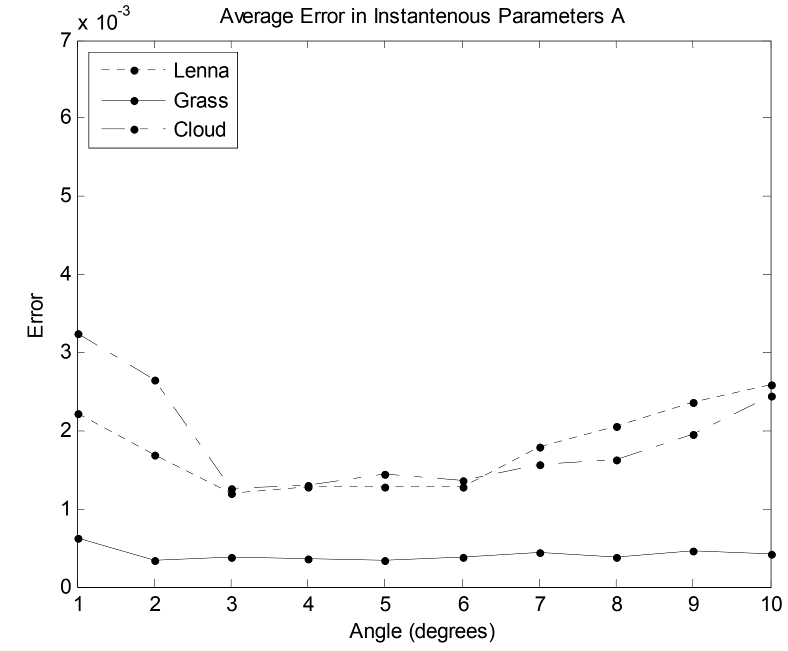


**Figure 6.58** Test 6 CWT – Rotation (**A**): Average error in the instantaneous parameters $p_1$ – $p_4$ for the Rotating Test sequence for Lenna, Grass, and Cloud. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

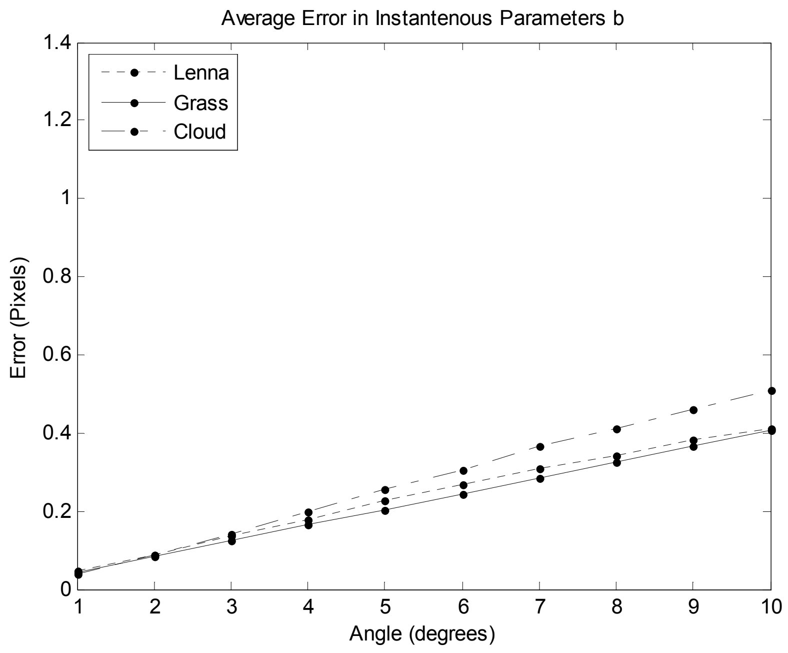


**Figure 6.59** Test 6 CWT – Rotation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Rotating Test sequence for Lenna, Grass, and Cloud. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

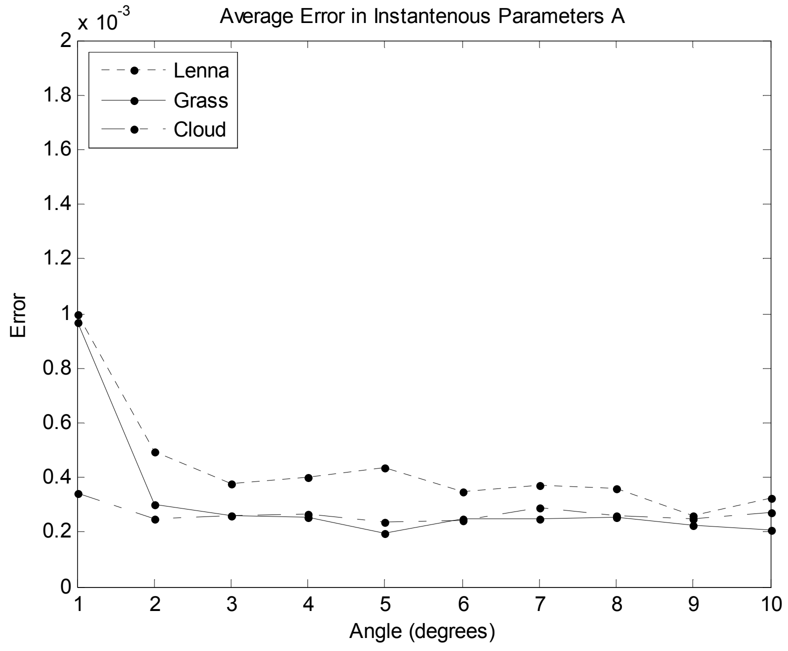


**Figure 6.60** Test 6 LK – Rotation (**A**): Average error in the instantaneous parameters $p_1 - p_4$ for the Rotating Test sequence for Lenna, Grass, and Cloud. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

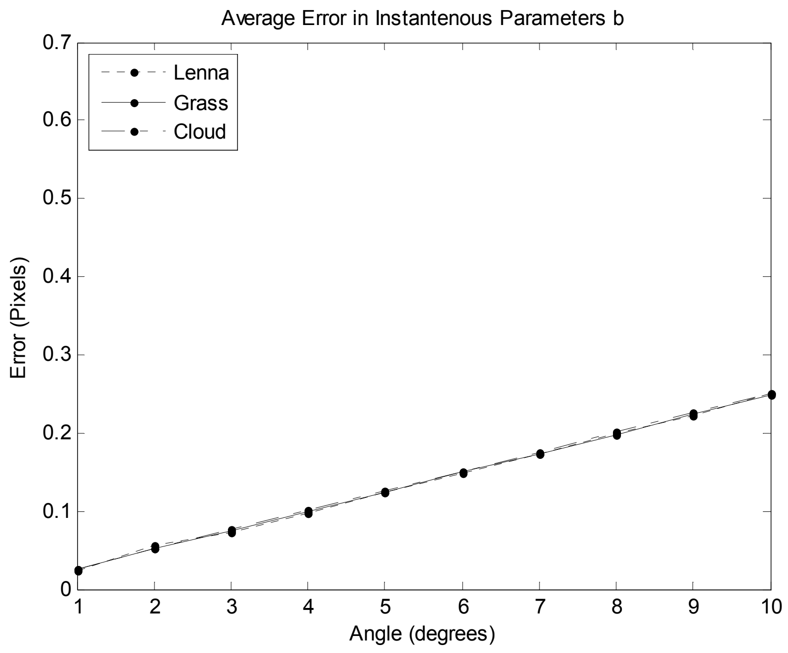


**Figure 6.61** Test 6 LK – Rotation (**b**): Average error in the instantaneous parameters $p_5$ and $p_6$ for the Rotating Test sequence for Lenna, Grass, and Cloud. The errors are computed for rotation angles 1° – 10°. The average is computed over 25 frames.

#### 6.2.8.2 Effect of Intensity Patterns to Illumination Change Robustness

The purpose of these tests is to examine the change in the behavior of the algorithm under illumination changes for different types of intensity patterns, or textures.

Two different tests are performed for the evaluations: additive offset (brightness) and intensity scaling (contrast). For brightness, an offset ranging from 0 to 25 is added to the sequences at each frame. For contrast, the intensity of the sequences is scaled by a factor ranging from 0.75 to 1.25.

The *affine* motion model with 64 × 64 pixels target size is used for the tests. The tests are all performed for the four types of motion. For the evaluations, the instantaneous error measures, given in Equations (6.12) and (6.13), are used. A comparison to Lucas-Kanade [57] method is also provided.

In order to simplify the graphics, we picked one representative case for each motion type: 4 pixels for translation, 5% for scale, 5% for shear, and 4° for rotation. The errors of each motion type are then averaged to obtain a single error value for each disturbance value. The results are given for brightness and contrast separately, for the Complex Wavelet Tracker first, and for Lucas-Kanade next from Figure 6.62 to Figure 6.69.

On the overall, it is observed that, even though there are differences in the error levels, for both algorithms the behavior is similar for varying textures. For the Complex Wavelet Tracker the accuracy is not affected by brightness and contrast changes for all three series. For the Lucas-Kanade, an increase in the errors is observed with increasing intensity perturbations for all three series; especially for the Cloud series, the increase in the error is quite high. Both behaviors are consistent with our expectations. Phase-based methods tend to be robust to intensity perturbations in contrast to intensity-based methods. The Complex Wavelet Tracker takes its advantage of being phased for these tests.

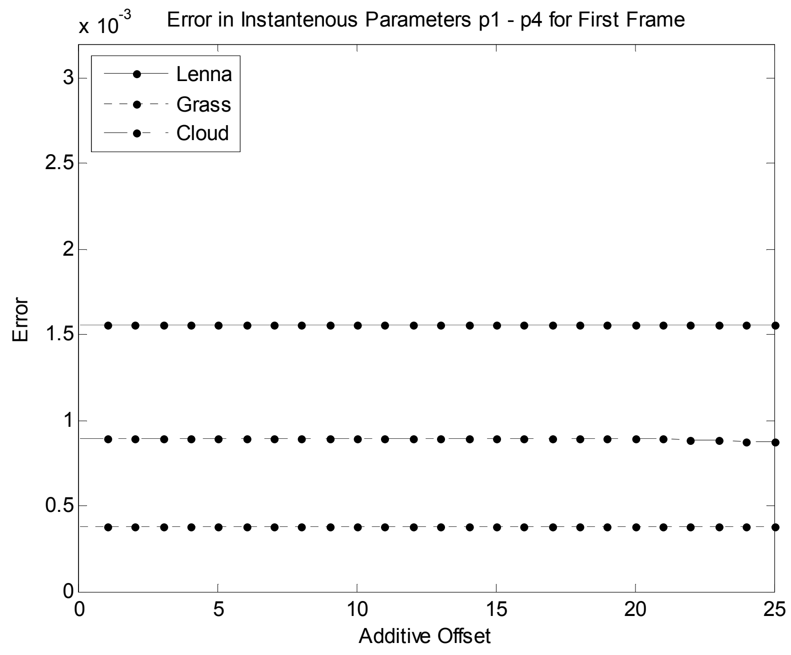


**Figure 6.62** Test 6 CWT – Brightness (**A**): Texture robustness to brightness changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

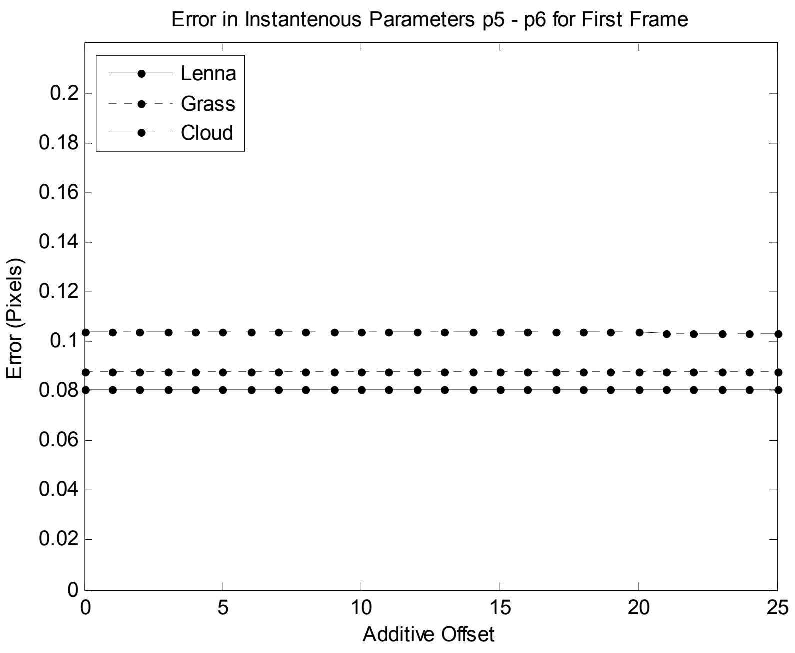


**Figure 6.63** Test 6 CWT – Brightness (**b**): Texture robustness to brightness changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

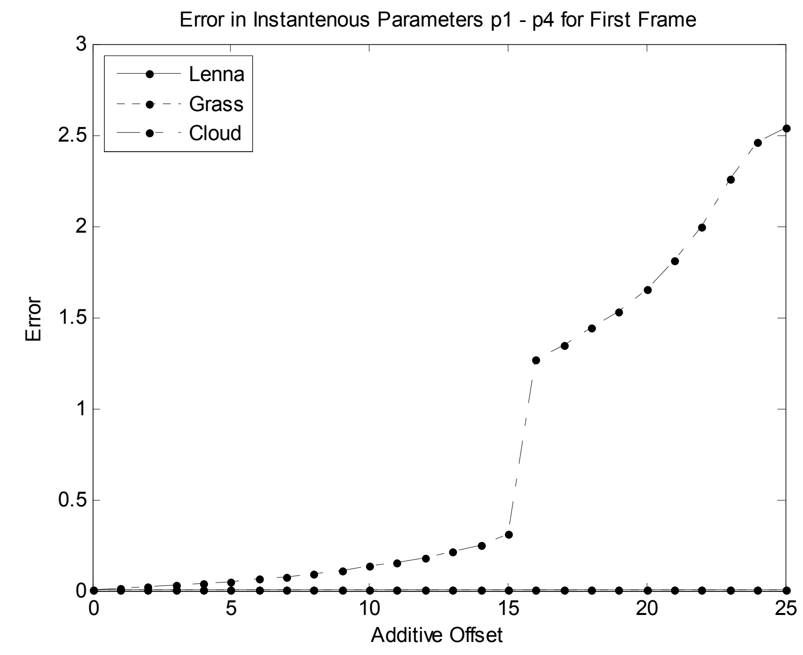


**Figure 6.64** Test 6 LK – Brightness (**A**): Texture robustness to brightness changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

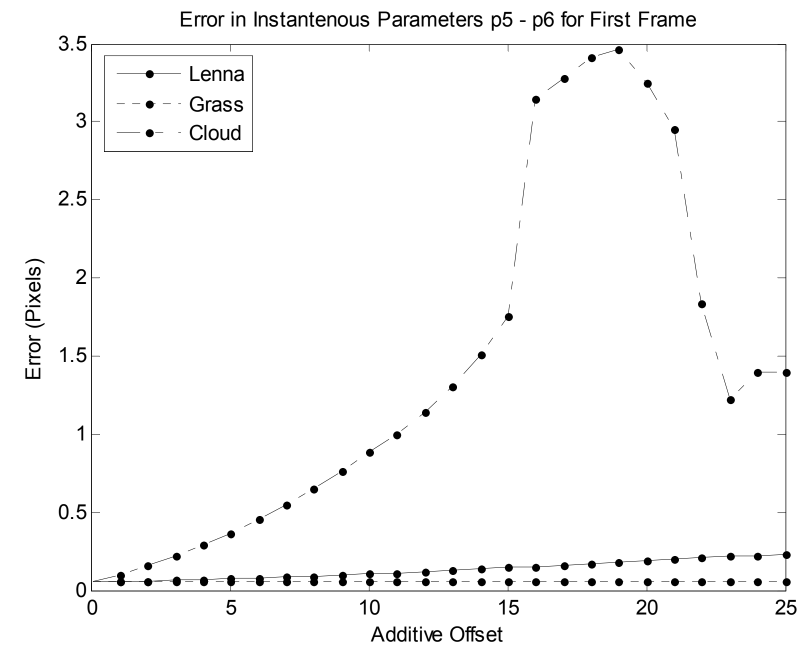


**Figure 6.65** Test 6 LK – Brightness (**b**): Texture robustness to brightness changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

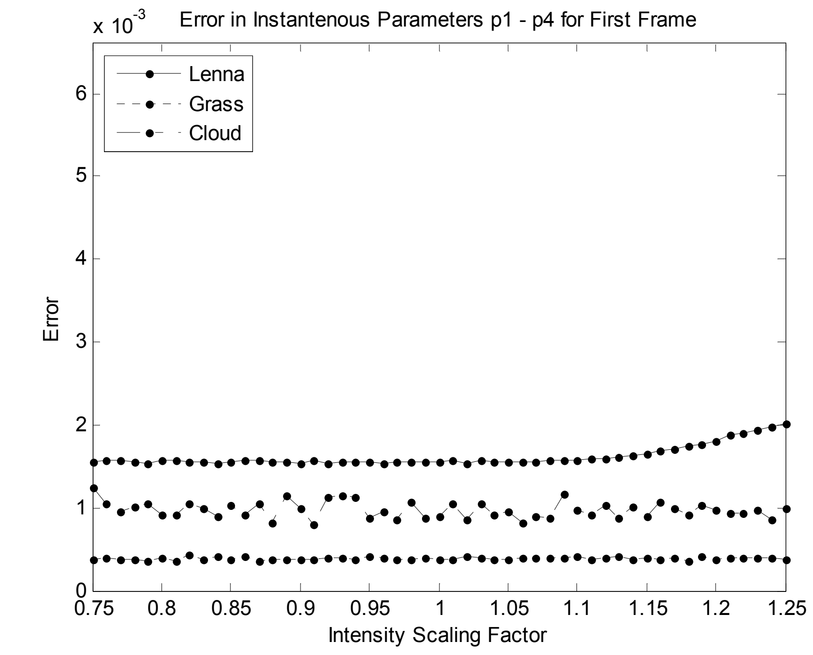


**Figure 6.66** Test 6 CWT – Contrast (**A**): Texture robustness to contrast changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

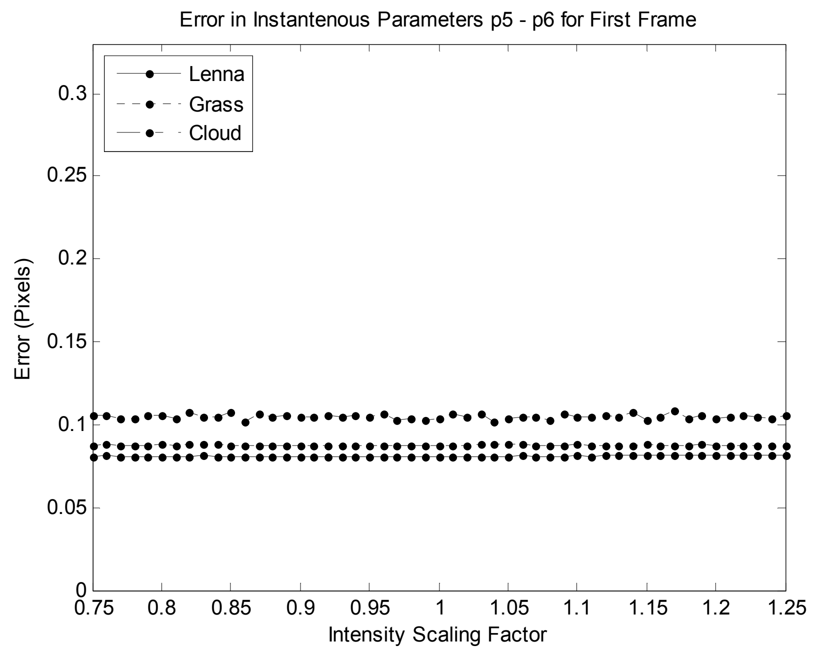


**Figure 6.67** Test 6 CWT – Contrast (**b**): Texture robustness to contrast changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

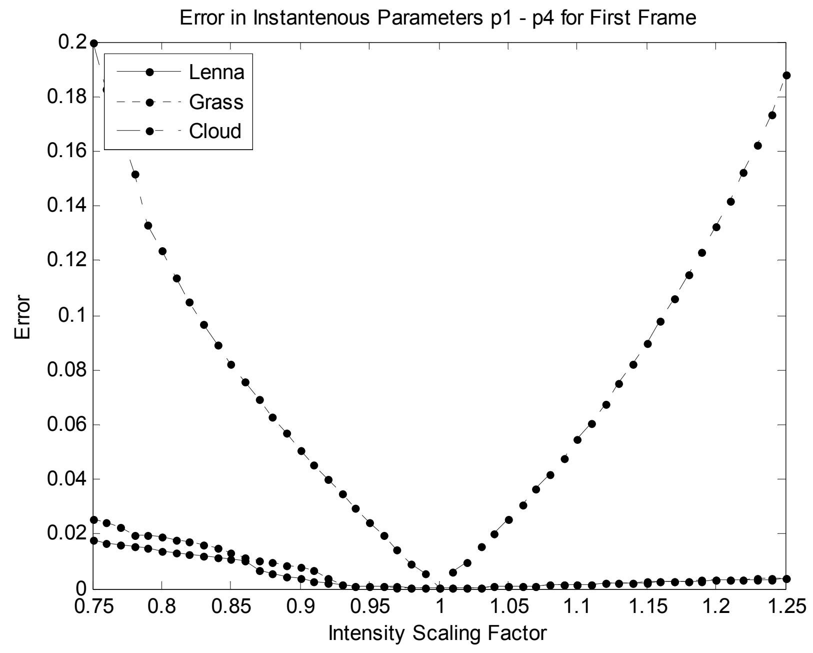


**Figure 6.68** Test 6 LK – Contrast (**A**): Texture robustness to contrast changes. The error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

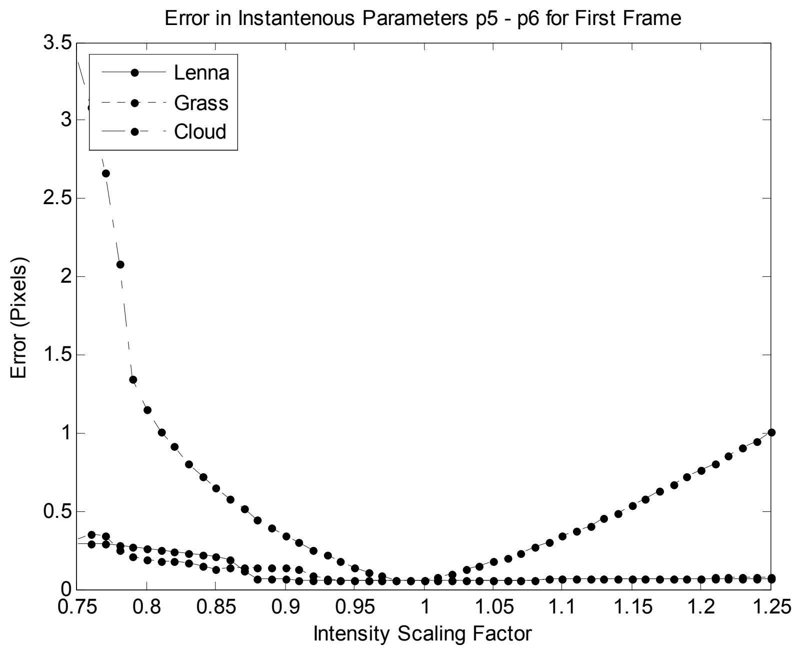


**Figure 6.69** Test 6 LK – Contrast (**b**): Texture robustness to contrast changes. The error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

#### 6.2.8.3 Effect of Intensity Patterns to Noise Robustness

The purpose of these tests is to evaluate the change in the behavior of the algorithm under additive white Gaussian noise for different types of intensity patterns, or textures.

Additive zero-mean white Gaussian noise with standard deviation varying from 0 to 20 is added to the sequences. The *affine* motion model with 64 × 64 pixels target size is used to perform these tests. The tests are all performed for the four types of motion. For the evaluations, the average instantaneous error measures, given in Equations (6.14) and (6.15), are used. A comparison to Lucas-Kanade [57] method is also provided.

In order to simplify the graphics, we picked one special example for each motion type: 4 pixels for translation, 5% for scale, 5% for shear, and 4° for rotation. The errors of each motion type are then averaged to obtain a single error value for each noise level. The results are given for the Complex Wavelet Tracker and the Lucas-Kanade algorithm separately from Figure 6.70 to Figure 6.73.

On the overall, it is observed that the Complex Wavelet Tracker and the Lucas-Kanade algorithms behave similarly under noise. For both algorithms, with increasing noise the error is gradually increasing for all series and their performances are similar for all three series. Their performances for Lenna and Grass are the same with a small increase in the error with noise, whereas for the Cloud series, both have a gradually increase in the error. As a result, both methods behave similarly under noise for different types of textures. Both show low performance for smooth textures under noise.

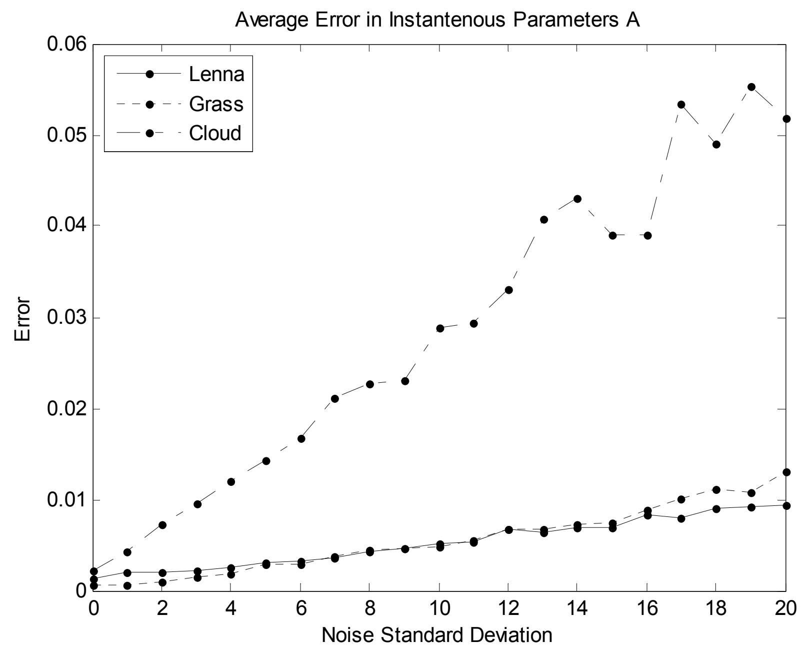


**Figure 6.70** Test 6 CWT – Noise (**A**): Texture robustness to noise. The average error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

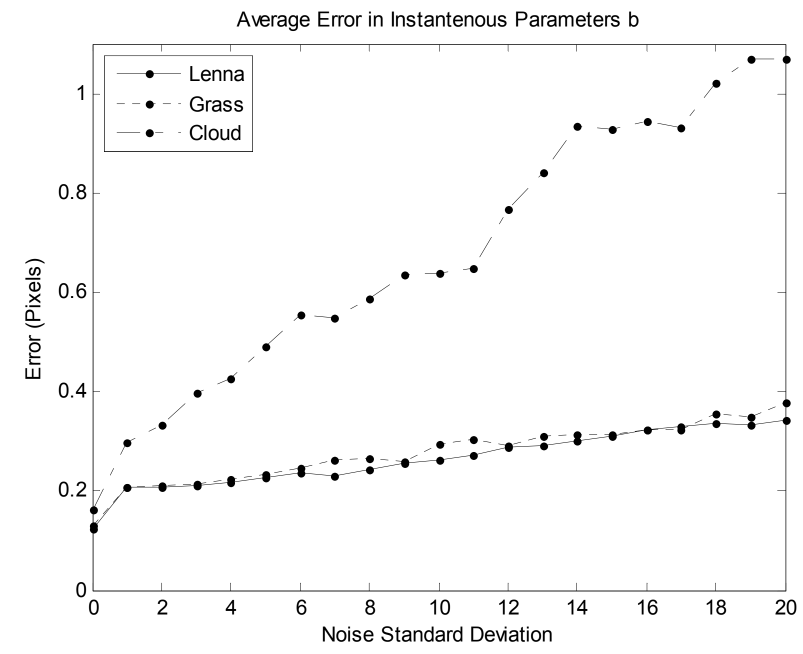


**Figure 6.71** Test 6 CWT – Noise (**b**): Texture robustness to noise. The average error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

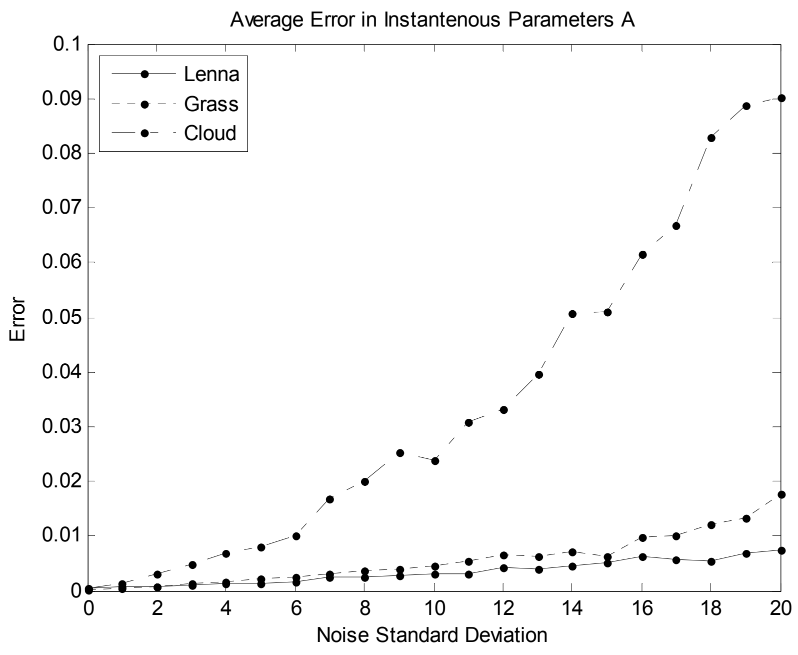


**Figure 6.72** Test 6 LK – Noise (**A**): Texture robustness to noise. The average error in the instantaneous parameters **A**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

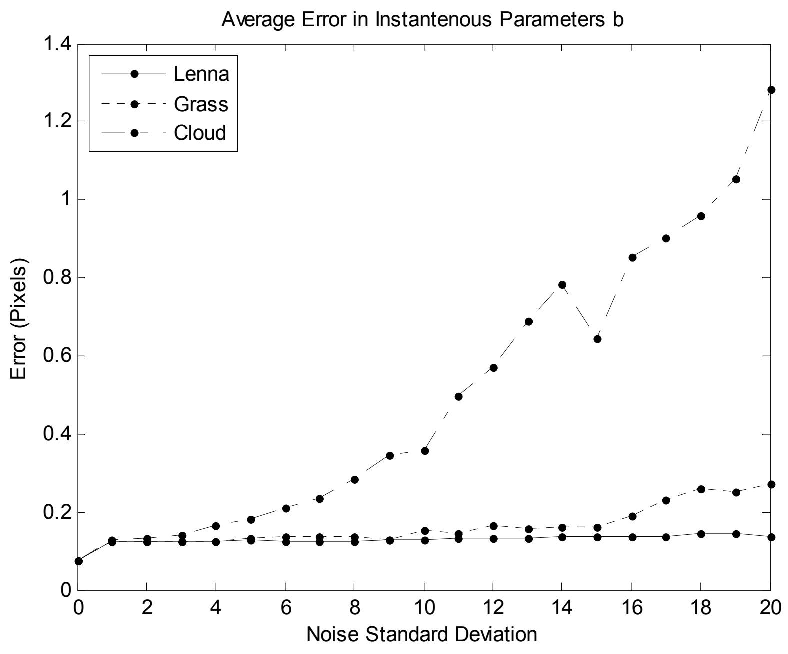


**Figure 6.73** Test 6 LK – Noise (**b**): Texture robustness to noise. The average error in the instantaneous parameters **b**, averaged from the errors for 4.00 pixel translation, 5% scaling, 5% shear, and 4° rotation.

#### 6.2.8.4 Summary of the Results

Tests have been performed to evaluate the change in the performance of the Complex Wavelet Tracker when the underlying texture that is tracked is being changed. Three types of textures have been used: Lenna, Grass, and Cloud. By varying the series, three types of robustness tests have been performed: First tests evaluate the change in the accuracy and affine limits with no added perturbations. Second and third tests evaluate the performance change under intensity perturbations and noise, respectively. The results have been compared with Lucas-Kanade [57].

The first group of test explores the change in the accuracy with respect to texture. It is observed that the accuracy remained similar within some limits for all series. Some changes in the limits have been observed that causes earlier or later breaks, changing the limits explored in Test 1 (Section 6.2.3) by 1-2 units. The overall performance is compatible with Lucas-Kanade.

The second group examines the change in the robustness under additive offset and intensity scaling. The Complex Wavelet Tracker proved its robustness for all three series by showing perfect immunity. Lucas-Kanade, however, being intensity based, showed several degrees of affect under these perturbations.

The last group of tests evaluates the change in the robustness to noise under varying texture. Same behavior is observed for both algorithms: with increasing noise, the accuracy is decreasing in a similar linear form.

As a result, it can be concluded that the Complex Wavelet Tracker performs approximately similar for different types of intensity patterns; hence, leading to a robust method under different types of targets.

### 6.2.9 Test 7: Random Affine Tests with Random Perturbations

The purpose of these tests is to evaluate the long-term performance under random affine motion with random perturbations. New test sequences of 1000 frames are created for the test. All three test images, Lenna, Grass, and Cloud are used for the sequences. The Lucas-Kanade [57] algorithm is used to compare the results.

The new test sequences are created in a similar way explained in Section 6.2.2. The motion parameters and the amount of perturbations, however, are obtained randomly for each frame in the sequence. For the affine parameters, 2 translation, 2 scale, 2 shear, and 1 rotation parameter is randomly generated for each frame. For the perturbations, again, 1 offset for brightness, 1 scale for contrast, and 1 standard deviation for noise are randomly generated for every frame. The random numbers are selected from a uniform distribution with limits set for each parameter. These limits are given in Table 6.4. So, for each frame, 10 random numbers have been used, making a total of 10.000 random numbers. The same random pattern is used for all sequences. Sample frames from the Lenna sequence are shown in Figure 6.74.

**Table 6.4** Limits for uniform random values for each deformation and perturbation parameter used to create the random sequences.

| *Parameter* | *Limits for Random Values* |
|---|---|
| Translation | –4 to +4 pixels for each axis |
| Scale | –2% to +2% for each axis |
| Shear | –2% to +2% for each axis |
| Rotation | –2° to +2° |
| Noise Standard Deviation | 10 |
| Additive Offset | ±10 |
| Intensity Scale Factor | 0.90 – 1.10 |

Four different test sequences are created with each image, making a total of 12 sequences.

The individual test cases are:

1. Random affine motion without any perturbations [Affine]
2. Random affine motion with random illumination change [Aff+Illum]
3. Random affine motion with random additive noise [Aff+Noise]
4. Random affine motion with both random illumination change and random additive noise [Aff+Ill+Noise]

The reason for these different test cases is to evaluate the effect of disturbances to the pure estimation accuracy.

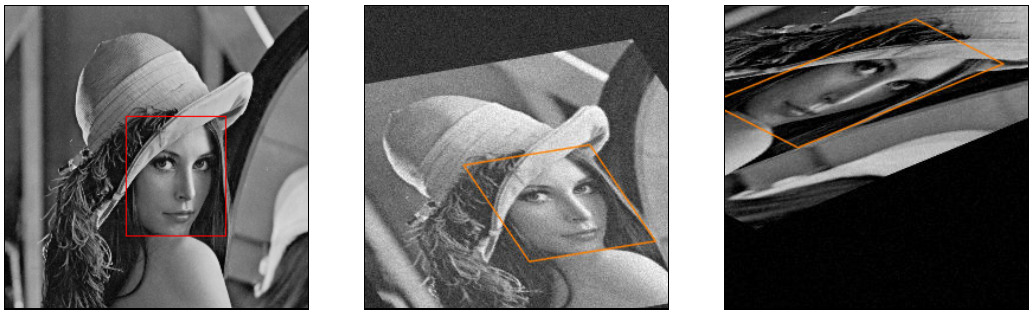


**Figure 6.74** Random Affine Test Sequence: A total of 1000 frames. Random affine motion, random noise and random illumination changes have been used to create the sequence. The size of the target selected on the first frame (left) is 85 × 102 pixels. The representative frames from left to right are 1st, 115th, and 1000th frame.

The *affine* motion model with a five-level pyramid ($m_{\max}$ = 5) is used to perform these tests. For the evaluations, the average instantaneous error measures, (6.14) and (6.15), the cumulative error measures, (6.16) and (6.17), and the track gate error measure (6.18) is used.

The results for all sequences are given in Figure 6.75 and Figure 6.76 showing the average instantaneous errors. For close inspection, the instantaneous errors together with the track gate error for the 4th Test Case of each sequence is given in Figure 6.77 through Figure 6.85.

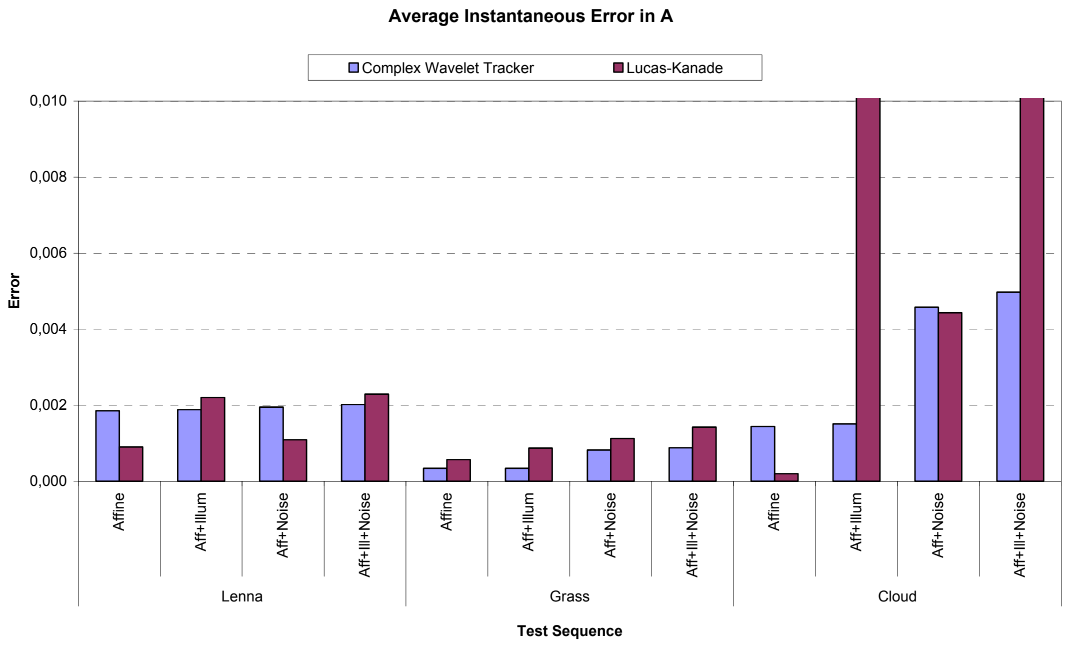


**Figure 6.75** Test 7 Random (**A**): Average error in the instantaneous parameters **A** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to Test Sequences.

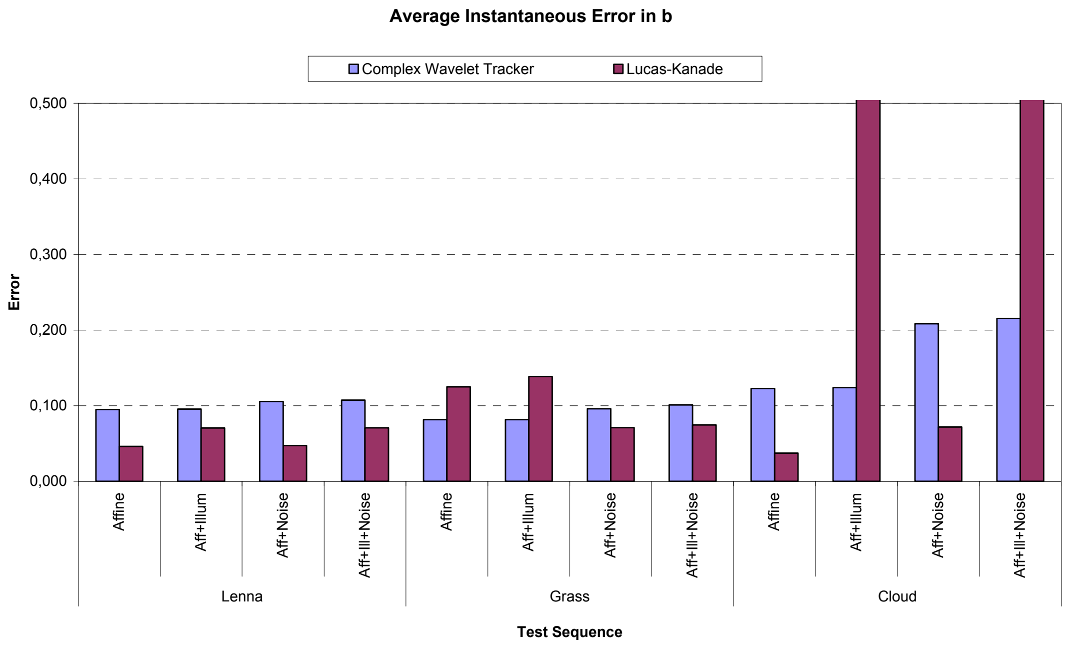


**Figure 6.76** Test 7 Random (**b**): Average error in the instantaneous parameters **b** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to Test Sequences

Examining first the results summarized in Figure 6.75 and Figure 6.76 we observe that for all sequences the Complex Wavelet Tracker remained in the accurate tracking region. The Lucas-Kanade, except for two cases, performed similarly and remained in the accurate region as well. For the two cases, which were the Cloud sequences having illumination changes present, the tracking broke down. The Cloud sequence is a very smooth image which is not suitable for intensity-based methods. Together with intensity perturbations, the high error for Lucas-Kanade is not surprising. The proposed method, in contrast, takes its advantage of being phase-based and shows its robustness on this difficult case as well.

For the Complex Wavelet Tracker, it is observed that it performed best for the Grass Sequences where the errors were below the Lucas-Kanade for all test cases. This shows that the Complex Wavelet Tracker operates more confidently on repetitive textures, or on texture-like targets. Comparing the errors of the 1st Test Case for Lenna and Cloud, we see that the error in the Cloud sequence is a bit smaller than the one in Lenna, which supports this argument when we take the Cloud image as being a texture-like surface as well.

For the Lenna Sequences, the performance of the Complex Wavelet Tracker was equal for all four test cases, showing perfect robustness to intensity perturbations and noise. The results for Lucas-Kanade for the same sequence were highly better for the 1st and 3rd Test Cases and slightly worse for the other two cases where illumination changes were present.

The worst performance for the Complex Wavelet Tracker was the 4th Test Case of the Cloud sequence followed by the 3rd Test Case, but still the errors are within the accurate tracking region. The high error is stimuated by additive noise. This is the same for the Cloud sequence where error increase due to noise is significant. Both sequences are texture-like, so that we can conclude that noise is decreasing the accuracy for texture-like targets. One has to note however that, even with noise, the accuracy of the Grass sequence is better than the Lenna without noise. So, the increase in the accuracy for repetitive texture-like regions is degraded with noise.

To summarize, it is observed that the proposed method proves itself to be quite robust under illumination changes, but is affected by noise to some degree. In contrast to Lucas-Kanade, which is more robust to noise, whereas is highly affected by illumination changes. This shows also the difference of phase-based and intensity-based-methods.

Further examination of the detailed cumulative parameter errors reveals that the Lucas-Kanade experienced several jumps in the error throughout the Grass and Cloud sequences. The cumulative errors for the Complex Wavelet Tracker, however, were very smooth in comparison, which again shows the advantage of the proposed method to tracking even under difficult circumstances.

As a result, it is observed that even if the accuracies are similar on the overall, the Complex Wavelet Tracker performed more robustly than the Lucas-Kanade for all test cases and proves to be suitable for tracking.

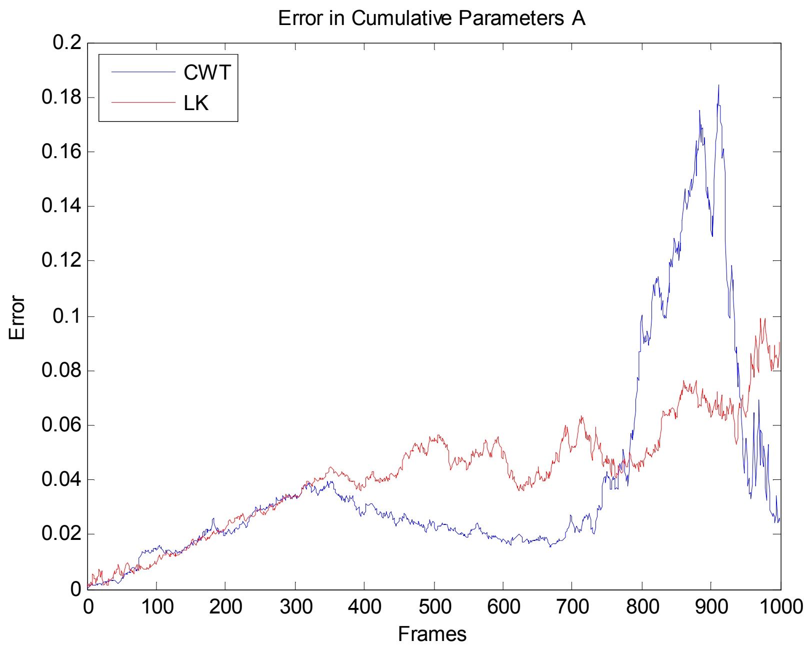


**Figure 6.77** Test 7 Random – Lenna (**A**): Lenna Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **A** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

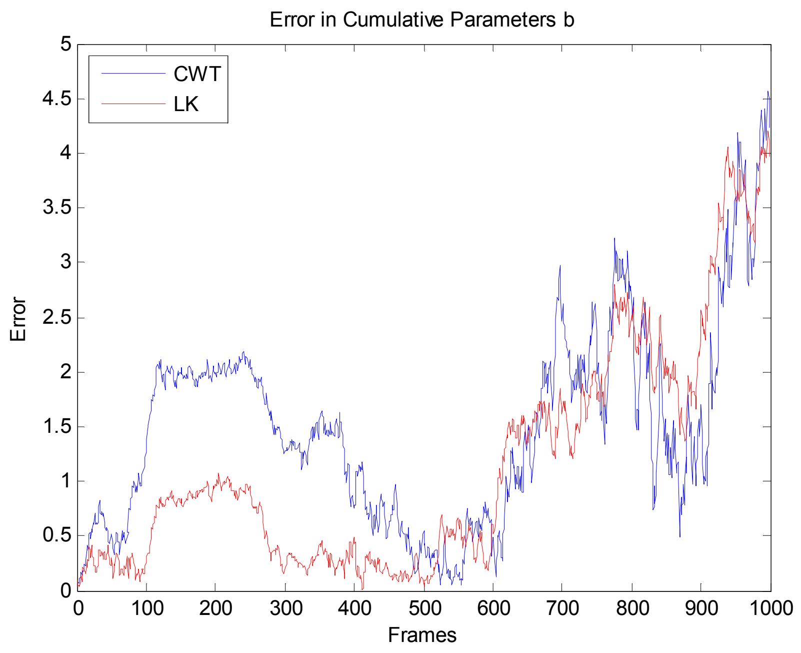


**Figure 6.78** Test 7 Random – Lenna (**b**): Lenna Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **b** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

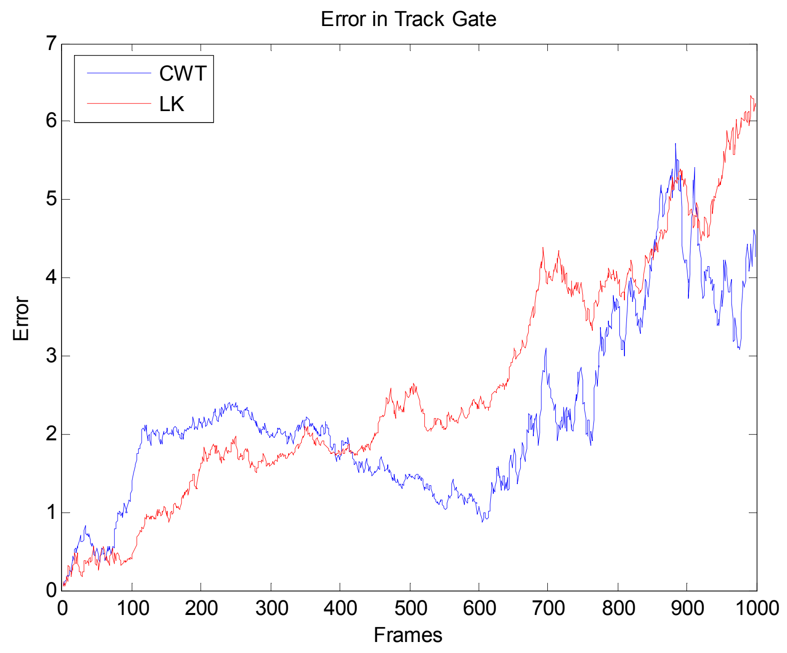


**Figure 6.79** Test 7 Random – Lenna (Track Gate): Lenna Random Test Sequence with random illumination and random noise. The error in the track gate for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

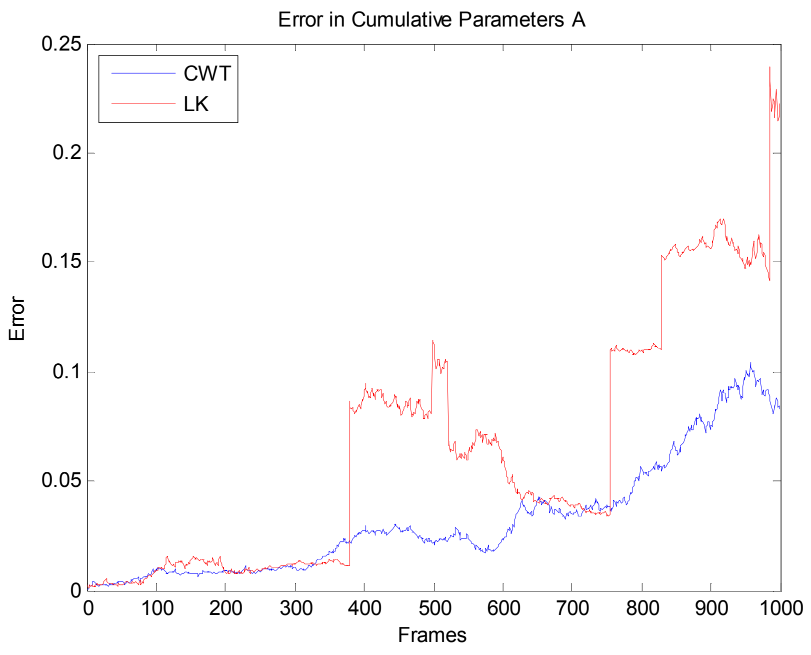


**Figure 6.80** Test 7 Random – Grass (**A**): Grass Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **A** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

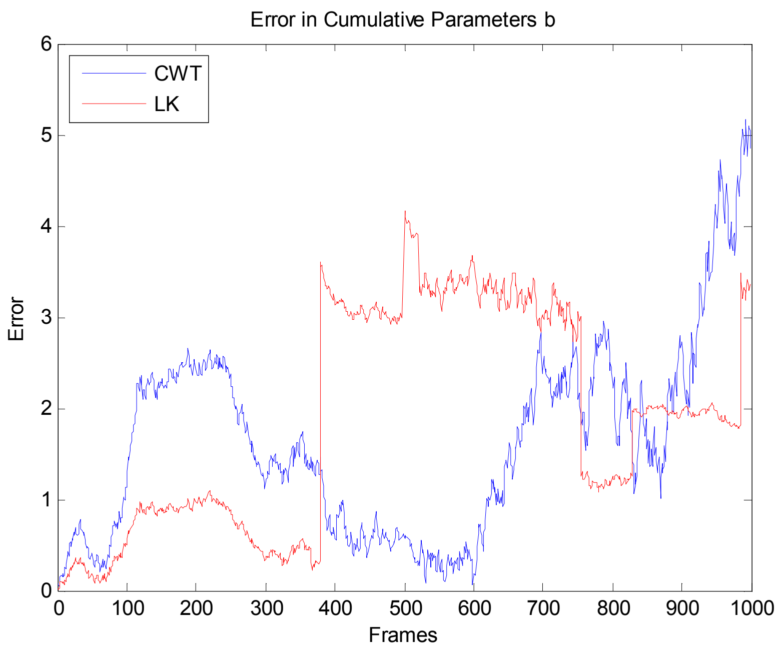


**Figure 6.81** Test 7 Random – Grass (**b**): Grass Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **b** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

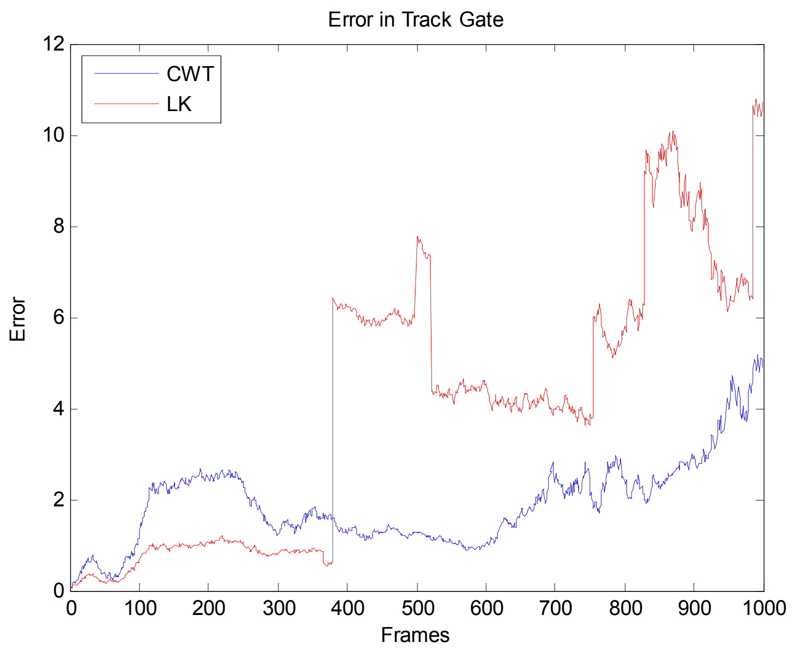


**Figure 6.82** Test 7 Random – Grass (Track Gate): Grass Random Test Sequence with random illumination and random noise. The error in the track gate for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

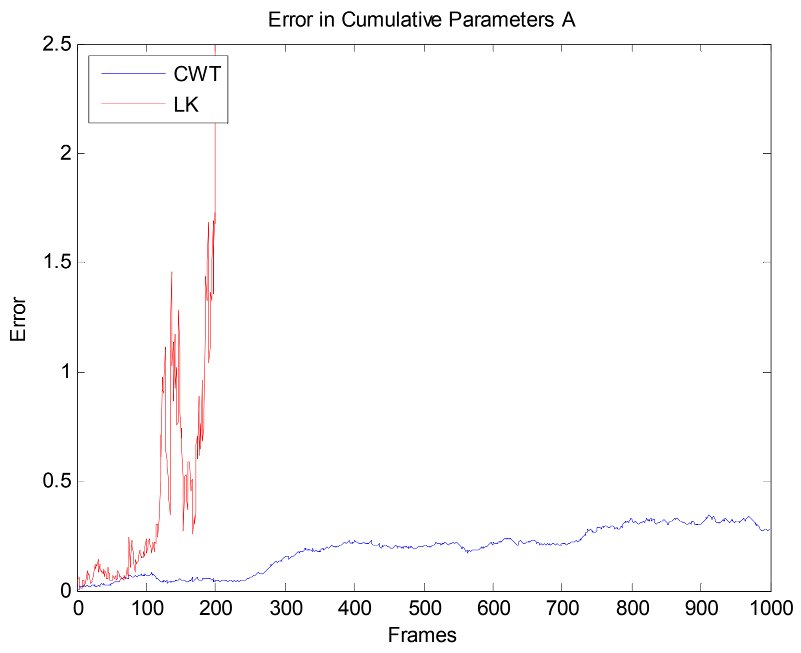


**Figure 6.83** Test 7 Random – Cloud (**A**): Cloud Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **A** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

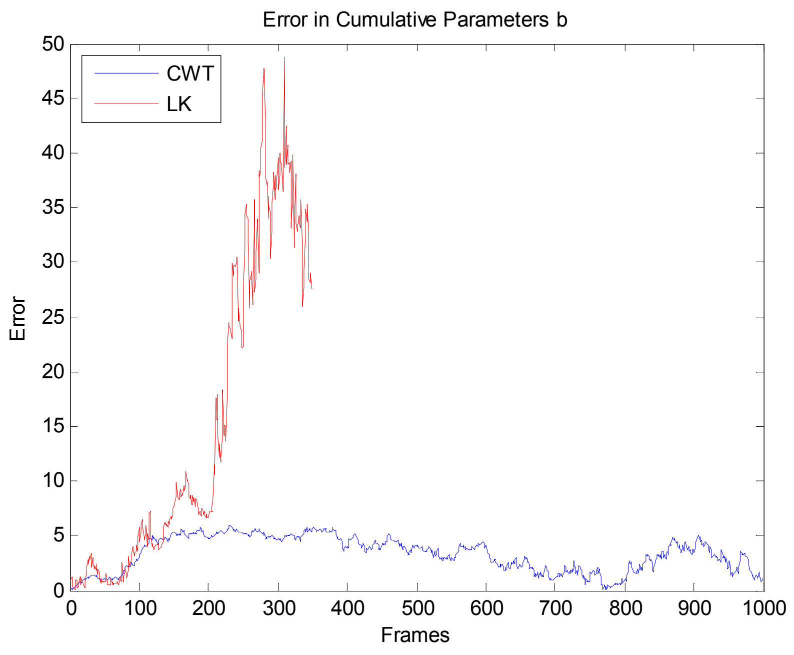


**Figure 6.84** Test 7 Random – Cloud (**b**): Cloud Random Test Sequence with random illumination and random noise. The error in the instantaneous parameters **b** for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

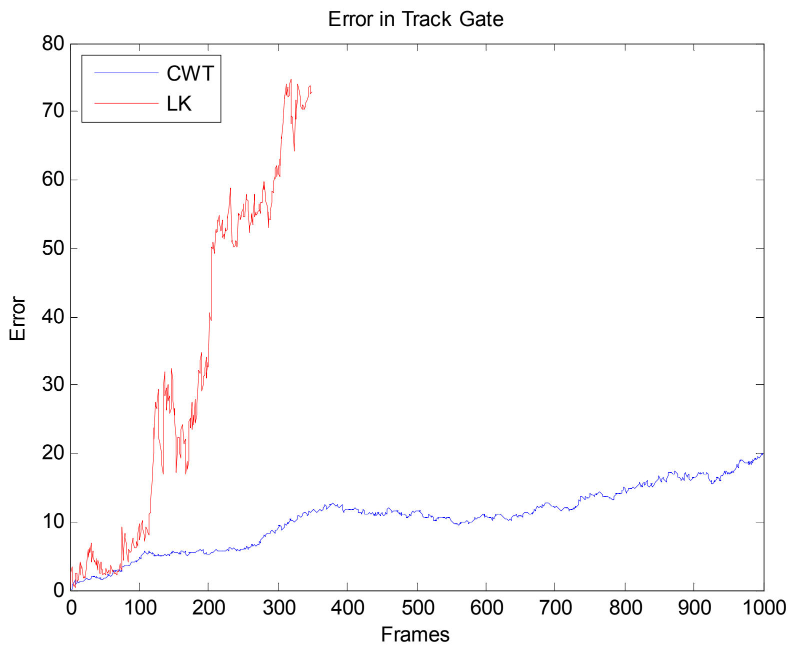


**Figure 6.85** Test 7 Random – Cloud (Track Gate): Cloud Random Test Sequence with random illumination and random noise. The error in the track gate for the Complex Wavelet Tracker (CWT) and Lucas-Kanade (LK) with respect to frames is shown.

## 6.3 Qualitative Tests: Tracking

Real sequences acquired from different imaging sources such as infrared, color and grayscale cameras are used for the qualitative evaluation of the proposed method. A wide area of applications is covered for the tests. The Lucas-Kanade [57] and the Mean Shift Tracker [24] are used for comparison.

We used the Lucas-Kanade's image registration method [57] with the *affine* motion model in the same way as the Complex Wavelet Tracker in order to have direct comparison of the two methods (See Section 0 for further explanations). The Mean Shift Tracker is implemented according to [24] and the target size adaptation method is performed as explained in [23]. The values used for the parameters for each sequence are given in the corresponding sections.

### 6.3.1 Test 1: Air02 Color Sequence

The first sequence is a color movie where a fighter aircraft maneuvers extensively in front of a blue sky background. The frames are 160 × 120 pixels and the sequence consists of 681 frames. Representative frames from the tracking sequence are shown in Figure 6.86 for the Complex Wavelet Tracker.

The Complex Wavelet Tracker managed to track the target quite successfully until the 530th frame. The target selected on the first frame is 39 × 27 pixels. The size grows as large as 65 × 65 pixels and the track finally breaks when the size reduces to 16 × 24 pixels. The target size at the final frame before the break occurs is 9 × 7 pixels. The small size in combination with a decrease in the contrast caused the break to occur.

For the same sequence, the Lucas-Kanade started in a similar way, but lost track at the 190th frame which is shown in Figure 6.87. Possibly due to low contrast, the gate size started to shrink around the 150th frame and eventually diminished as shown in the right-most frame of Figure 6.87.

The Mean Shift Tracker, however, maintained the track throughout the full sequence, although the gate is not perfectly matched. Representative frames from the Mean

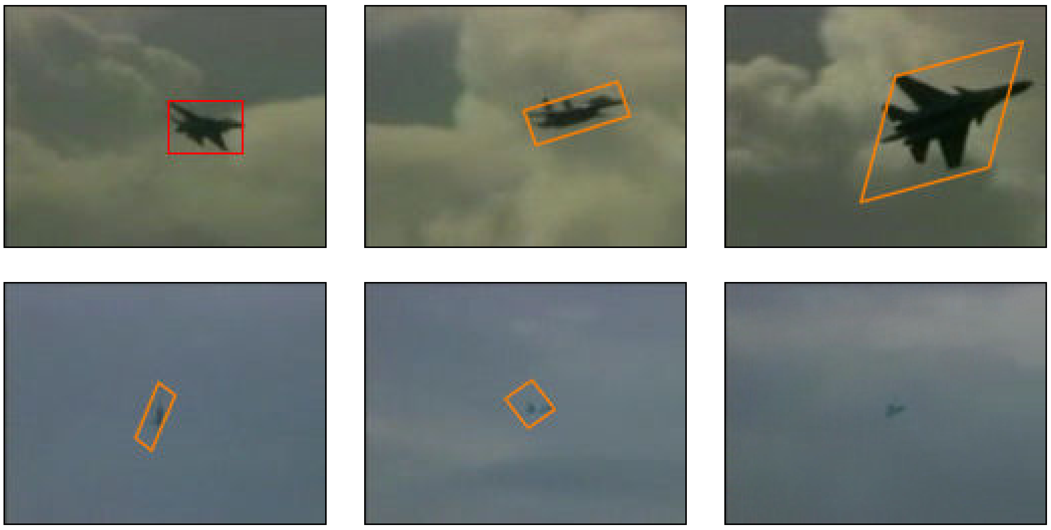


**Figure 6.86** Test 1 CWT – Air02 Sequence: A total of 681 frames. The size of the target selected on the first frame (upper left) is 39 × 27 pixels. The representative frames from left to right and top to bottom are 1st, 25th, 67th, 205th, 530th and 681st. The target is tracked successfully until the 530th frame.

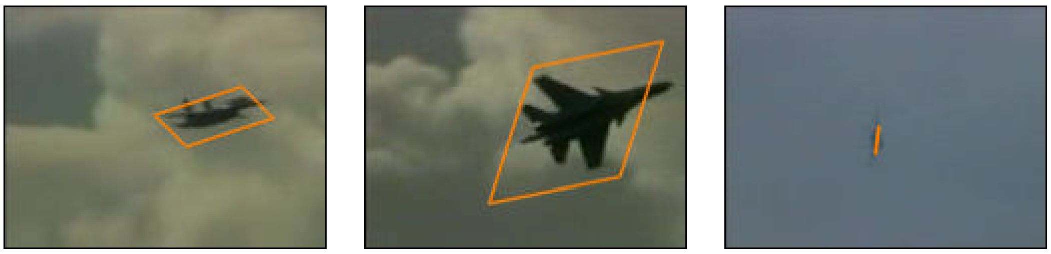


**Figure 6.87** Test 1 LK – Air02 Sequence: Lucas-Kanade managed to track successfully until the 190th frame which is shown on the right. The other two frames are the 25th (left) and 67th (middle).

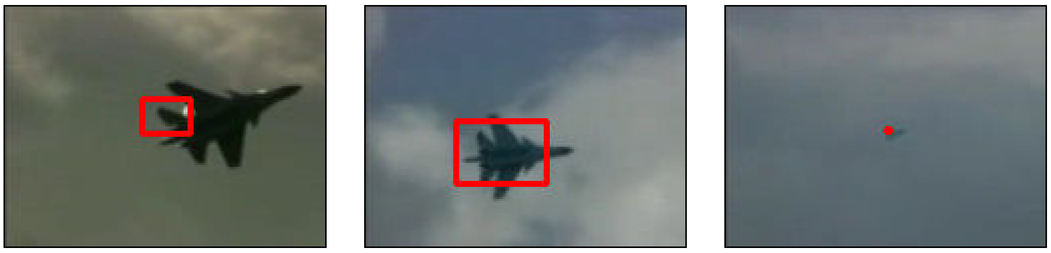


**Figure 6.88** Test 1 MS – Air02 Sequence: The Mean Shift Tracker managed to track the target successfully until the last frame. Representative frames from the tracking sequence are shown, which are from left to right, the 59th, 123rd, and 681st (last frame).

Shift Tracker is given in Figure 6.88. The parameters for the Mean Shift Tracker were set as follows: Number of Bins = 64, Size Ratio = 30%, Size Update = 0.05.

### 6.3.2 Test 2: Air08 Color Sequence

The second sequence is a color movie where a light aircraft maneuvers extensively in front of a blue sky background. The frames are 160 × 120 pixels and the sequence consists of 748 frames. Representative frames from the tracking sequence are shown in Figure 6.89 for the Complex Wavelet Tracker.

The Complex Wavelet Tracker managed to track the target successfully throughout the whole sequence. The target size on the first frame is 19 × 35 pixels. The track gate grows as large as 65 × 61 pixels and shrinks as small as 13 × 20 pixels throughout the sequence.

For the same sequence, the Lucas-Kanade started in a similar way, but lost track after the 411th frame which is shown in Figure 6.90. The small target size and low contrast caused difficulties for the method and starting from around the 350th frame, the gate adaptation was not confident and eventually diverged at the 411th frame. In the same period, the Complex Wavelet Tracker shows a quite confident adaptation of the gate and survives this difficult case.

The Mean Shift Tracker, maintained the track throughout the whole sequence, although the gate was not perfectly maintained. Representative frames are shown in Figure 6.91. The parameters for the Mean Shift Tracker were set as follows: Number of Bins = 128, Size Ratio = 5%, Size Update = 0.25.

### 6.3.3 Test 3: Air09 Color Sequence

Another color movie is used for the third test where a light aircraft maneuvers extensively in front of a blue sky background. The frames are 352 × 288 pixels and the sequence consists of 887 frames. Representative frames from the tracking sequence are shown in Figure 6.92 for the Complex Wavelet Tracker.

The Complex Wavelet Tracker maintains that track gate on the target until the last frame. The target size on the first frame is 31 × 33 pixels. The track gate grows as large as 228 × 144 pixels.

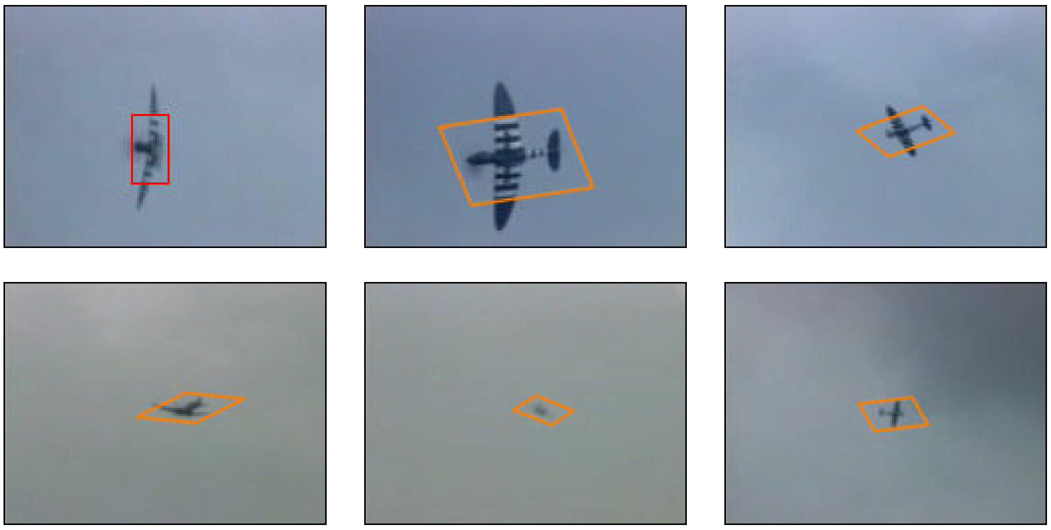


**Figure 6.89** Test 2 CWT – Air08 Sequence: A total of 748 frames. The size of the target selected on the first frame (upper left) is 19 × 35 pixels. The representative frames from left to right and top to bottom are 1st, 52nd, 148th, 241st, 524th and 748th. The target is tracked successfully until the last frame.

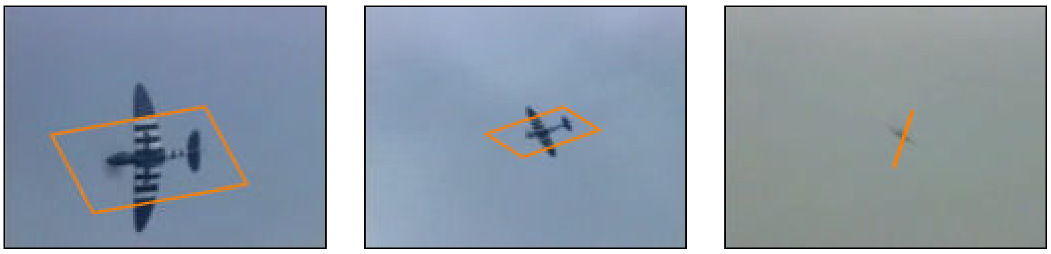


**Figure 6.90** Test 2 LK – Air08 Sequence: Lucas-Kanade managed to track successfully until the 411th frame which is shown on the right. The other two frames are the 52nd (left) and 148th (middle).

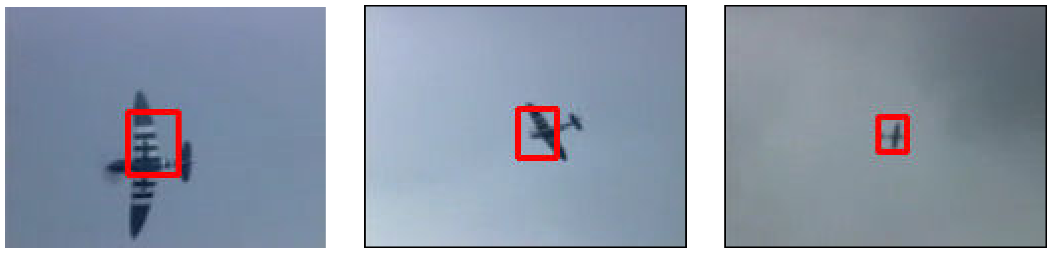


**Figure 6.91** Test 2 MS – Air08 Sequence: The Mean Shift Tracker managed to track the target successfully until the last frame. Representative frames from the tracking sequence are shown, which are from left to right, the 42nd, 160th, and 748th (last frame).

For the same sequence, the Lucas-Kanade started in a similar way. However, the track gate started to grow after the 140th frame and continued growing until the 220th frame, from which on, even though the gate was updated accordingly, the size was too large to handle and eventually crossed frame boundaries and the track was broken.

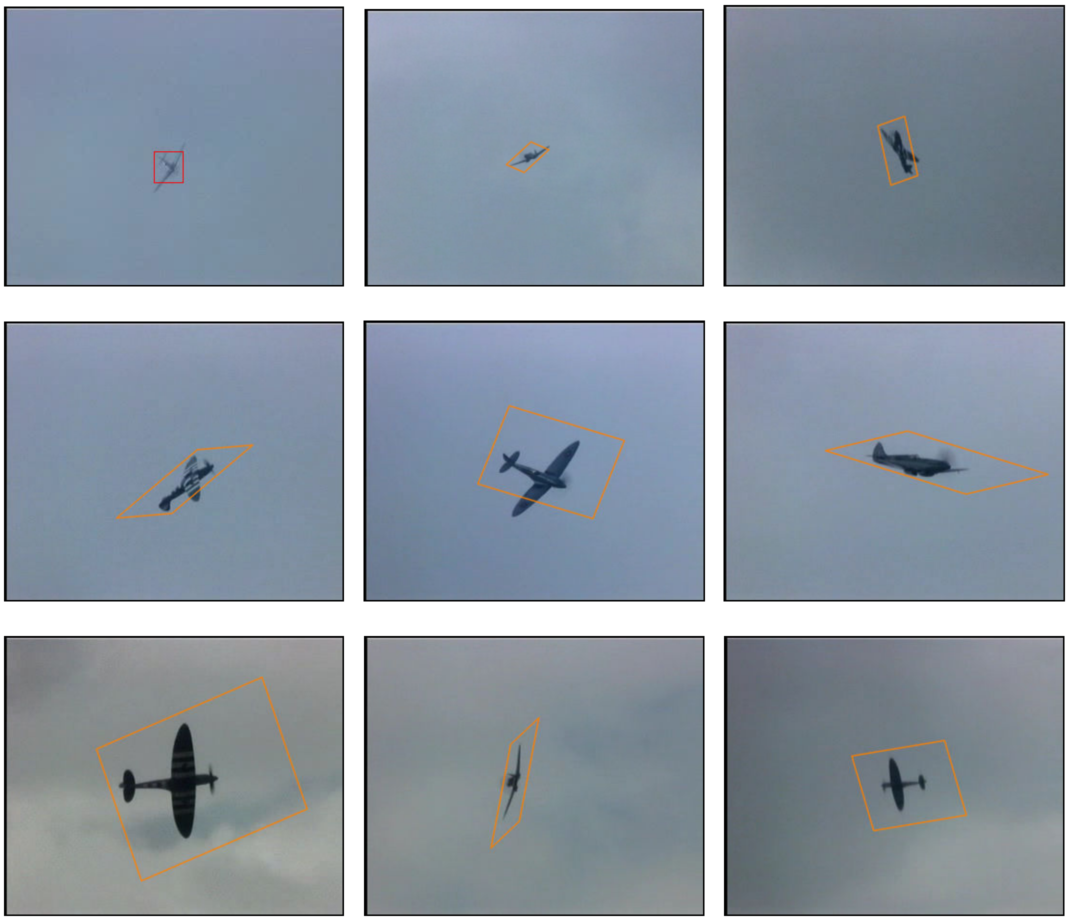


**Figure 6.92** Test 3 CWT – Air09 Sequence: A total of 887 frames. The size of the target selected on the first frame (upper left) is 31 × 33 pixels. The representative frames from left to right and top to bottom are 1st, 102nd, 220th, 319th, 478th, 552nd, 623rd, 713th, and 887th. The target is tracked successfully until the last frame.

The sequence starts with low contrast and small target size, which causes a small drift of the gate right at the initial period of the track. Due to this drift, in addition to a series of false updates after the 140th frames, which are possibly effected from the background due to the large gate size in comparison to the target size, eventually leads to a loss of the target. In the same periods, the Complex Wavelet Tracker updates the gate quite confidently and overcomes this situation successfully. Representative frames for the Lucas-Kanade are shown in Figure 6.93.

The Mean Shift Tracker maintained to track the target until the 627th frame. The tracker could not manage to update the track gate appropriately after the target became larger and had more contrast, thus making the match to the initial target model difficult and hence, a break occurs at the end. Representative frames from the Mean Shift Tracker are given in Figure 6.94. The parameters for the Mean Shift Tracker were set as follows: Number of Bins = 128, Size Ratio = 5%, Size Update = 0.25.

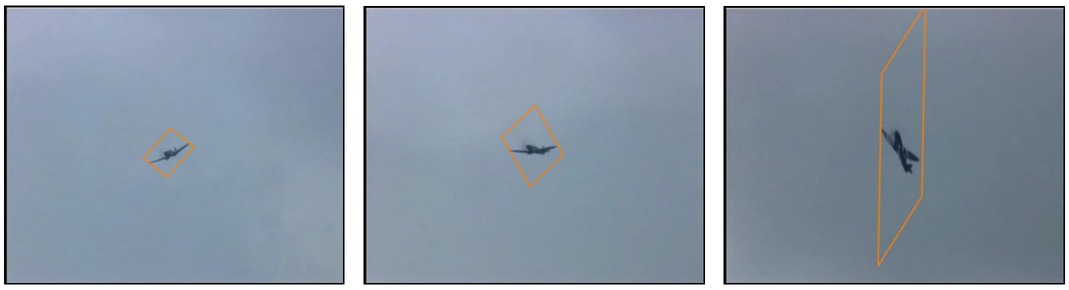


**Figure 6.93** Test 3 LK – Air09 Sequence: Lucas-Kanade managed to track successfully until the 140th frame which is shown in the middle. From then on the track gate started to grow more than necessary. The other two frames are the 102nd (left), and 220th (right).

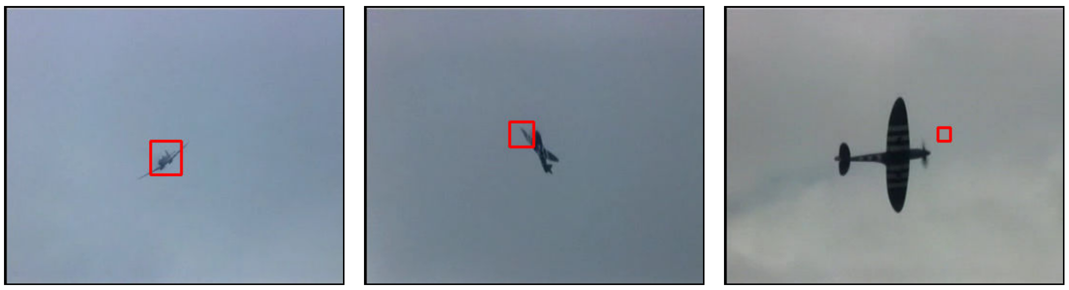


**Figure 6.94** Test 3 MS – Air09 Sequence: The Mean Shift Tracker managed to track the target successfully until the 627th frame which is shown on the right. Representative frames from the tracking sequence are shown, which are from left to right, the 70th, 319th, and 627th (last frame).

### 6.3.4 Test 4: Pursaklar03 Infrared Sequence

The fourth sequence is taken from an infrared camera. A public bus is approaching the camera. There is a highly complex background behind the target. The contrast is low and the images are not sharp. Similar textures are present in target and background. The frames are 320 × 240 pixels. Representative frames from the tracking sequence are shown in Figure 6.95 together with the gate updated by the Complex Wavelet Tracker.

The Complex Wavelet Tracker manages to track the target until the 269th frame. Although the scaling of the gate remained a bit less than the actual scaling of the target, the overall gate remained on the target with a slight drift in the target center.

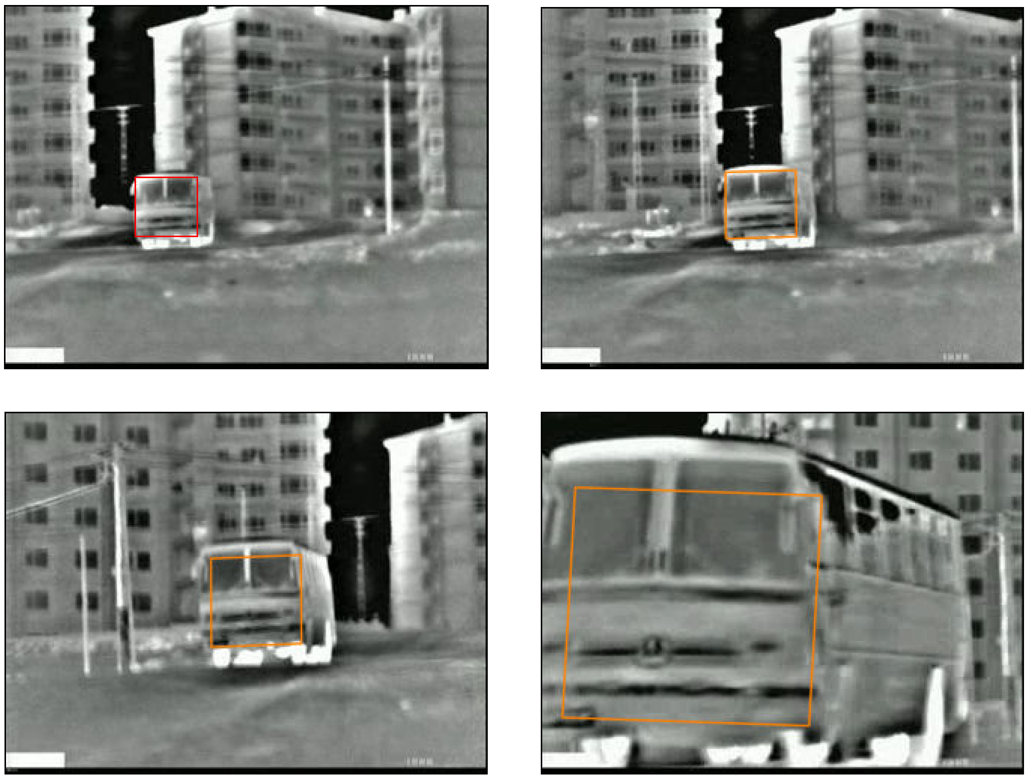


**Figure 6.95** Test 4 CWT – Pursaklar03 Infrared: The tracking sequence is 269 frames. The representative frames show the 1st, 42nd, 138th and 269th (last) frame of the sequence.

For the same sequence, the Lucas-Kanade performed in a similar way. The adaptation of the target size was better than the Complex Wavelet Tracker. The target center was also maintained more appropriately. This superiority might be the advantage of being intensity-based where the intensities are matched appropriately in the spatial domain in contrast to the complex wavelet domain where a transformation is present and shift invariance is supplied only in the approximate sense. Representative frames from the tracking sequence for Lucas-Kanade are shown in Figure 6.96.

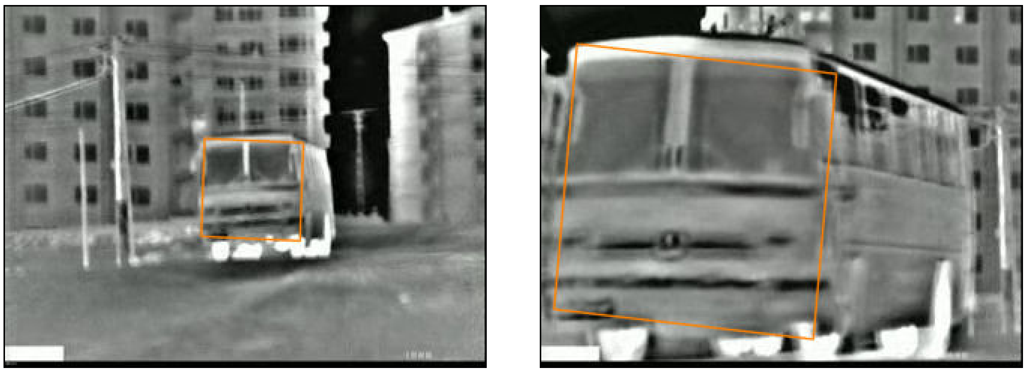


**Figure 6.96** Test 4 LK – Pursaklar03 Infrared: Lucas-Kanade managed to track successfully until the last frame which is shown on the right. The representative frames are the 138th (left) and 268th (right).

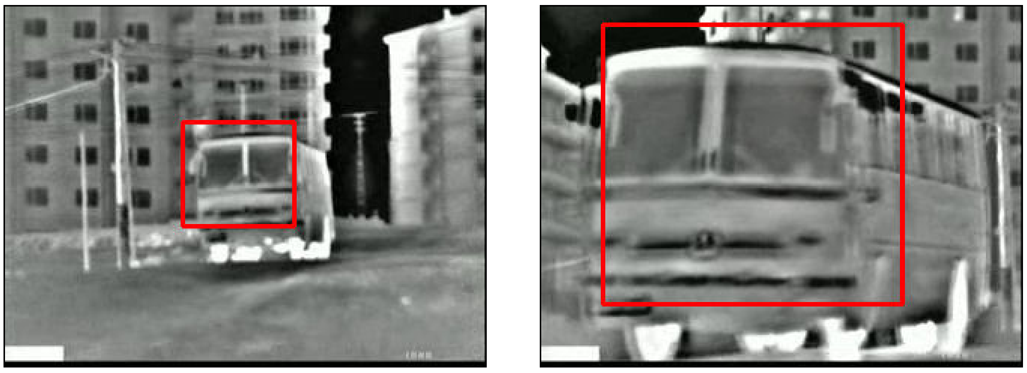


**Figure 6.97** Test 4 MS – Pursaklar03 Infrared: The Mean Shift Tracker managed to track the target successfully until the last frame. Representative frames from the tracking sequence are the 138nd (left) and 267th (right).

The Mean Shift Tracker maintained the track throughout the full sequence, although drifts and mismatches occurred during the track. Representative frames from the Mean Shift Tracker are given in Figure 6.97. The parameters for the Mean Shift

Tracker were set as follows: Number of Bins = 64, Size Ratio = 30%, Size Update = 0.05.

### 6.3.5 Test 5: Pursaklar03 Color Sequence

The fifth sequence is the same scene as the previous one, this time acquired by a color camera. A public bus is approaching the camera. The frames are 352 × 288 pixels. Representative frames from the tracking sequence are shown in Figure 6.98 for the Complex Wavelet Tracker.

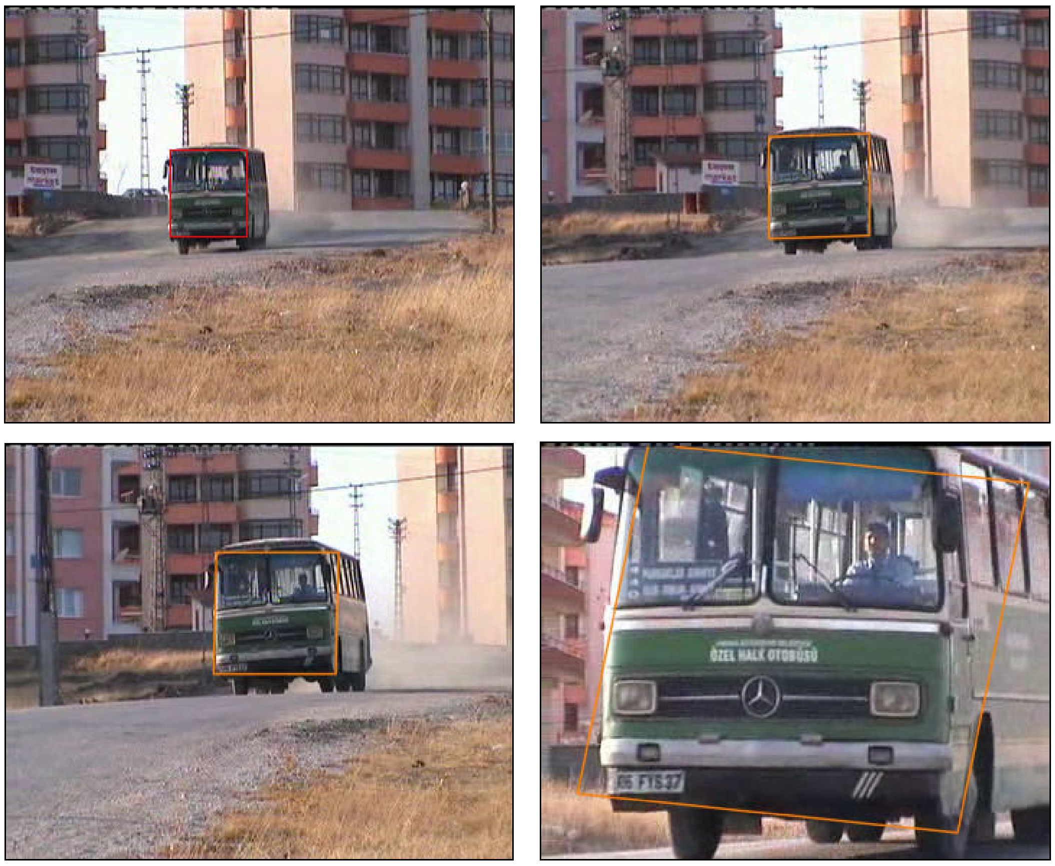


**Figure 6.98** Test 5 CWT – Pursaklar03 Color Sequence: The tracking sequence is 226 frames. The representative frames show the 1st, 52nd, 95th and 226th (last) frame of the sequence.

The target is tracked successfully until the last frame. The scaling of the gate is more appropriate than the one performed on the infrared sequence. However, for this case, the tilt at the beginning of the sequence caused an exaggerated rotation of the gate at the end of the sequence.

For the same sequence, the Lucas-Kanade started in a similar way. However, right at the beginning it lost the target after 16th frame by shrinking the gate to the left. This unexpected shrinking starts with the motion of the camera to left, which might be a possible cause. This is shown in Figure 6.99.

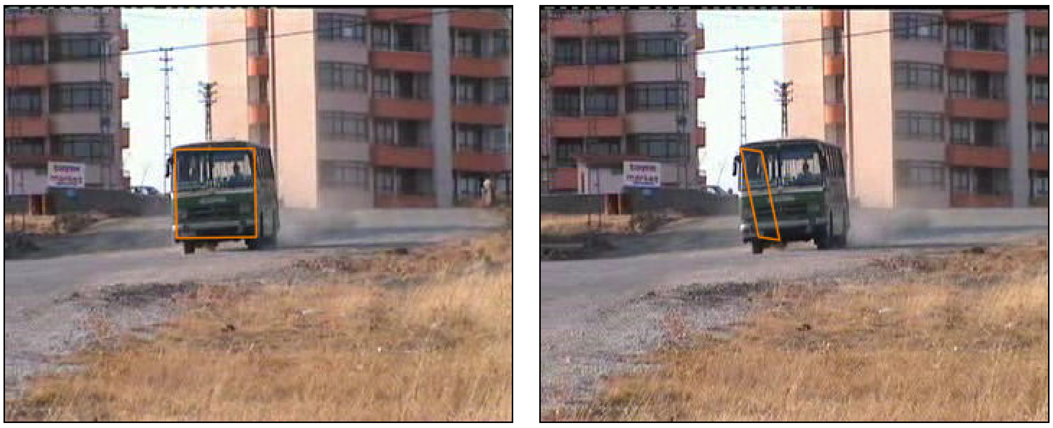


**Figure 6.99** Test 5 LK – Pursaklar03 Color: Lucas-Kanade managed to track successfully until the 22nd frame which is shown on the left and suddenly lost the target. The representative frames are the 16th (left) and 22nd (right).

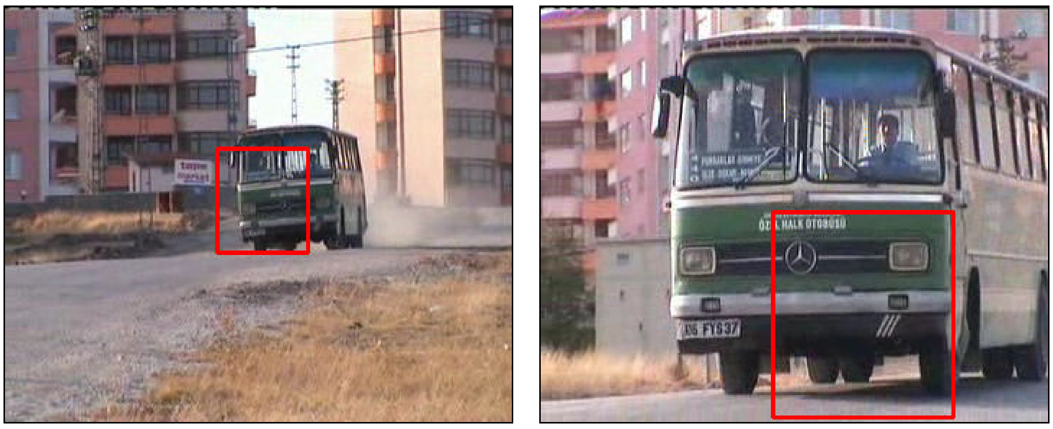


**Figure 6.100** Test 5 MS – Pursaklar03 Color: The Mean Shift Tracker managed to track the target until the 208th frame. It also had difficulties in maintaining the track gate on the target and updating the size accordingly. Representative frames from the tracking sequence are the 52nd (left) and 208th (right).

The Mean Shift Tracker, confronted difficulties in maintaining the gate on the target. It also experienced difficulties in updating the gate appropriately. Representative frames from the Mean Shift Tracker are given in Figure 6.97. The parameters for the Mean Shift Tracker were set as follows: Number of Bins = 64, Size Ratio = 30%, Size Update = 0.05.

### 6.3.6 Summary of the Results

Tracking simulations using real video sequence are performed. The results are compared with the Lucas-Kanade [57] and the Mean Shift Tracker [24]. The target is selected by the operator. It is observed that the proposed method is successful in tracking the selected area for both cluttered and uncluttered scenes.

For the air sequences it takes its advantage of being phase-based to overcome low contrast and illumination changes in comparison to intensity-based methods. For the pursaklar sequences, it performs compatible with Lucas-Kanade. The Mean Shift Tracker, although successful for most of the sequences, lacks accuracy and proper gate size adaptation which are important for the problems dealed in this work.

On the overall, it is observed that, for the test sequences, the Complex Wavelet Tracker performed better than the other two methods, in terms of both accuracy and robustness.

# CHAPTER 7

# DISCUSSION AND CONCLUSION

This Chapter starts with discussions on the Complex Wavelet Tracker and the simulation results. It continues with suggestions for further work and ends with a conclusion of the thesis.

## 7.1 Discussion on the Complex Wavelet Tracker

A new tracking algorithm that is based on the Complex Discrete Wavelet Transform (CDWT) [59] is proposed. The target is defined by a rectangular region which is then updated by the tracker according to a parametric motion model. The algorithm estimates the track parameters in the complex wavelet domain. The complex wavelet phase is used to match the wavelet coefficients of successive frames. In this way, the algorithm can also be viewed as a generalization of the CDWT based motion estimation method [59] developed by Magarey and Kingsbury. Instead of estimating the motion of a pixel, the motion of a region is estimated according to a parametric motion model.

The motion model can range from simple *translation* to *affine* motion, or, a subset of motion types can be defined in order to match the requirements of the specific application it is intended to be used for. We have investigated three types of motion models: the *translational*, the *similarity*, and the *affine* motion model. The solutions to these models are obtained as a linear equation solution in a least squares sense. Therefore, the selection of the motion model has little effect to the computational cost of the algorithm. The selection should be preferred rather to increase the performance of the tracking by preventing redundant degrees of freedoms of the track gate.

The Complex Wavelet Tracker is a generalization of the original motion estimation method of Magarey [59]. Increasing the support has lead to a more generic algorithm that has more a diverse use like block motion estimation, image registration, image alignment and tracking. The increase in the support has also a positive effect on robustness and accuracy.

The solution is obtained in a direct way, without the need for any iteration steps. This property leads to fixed computation times, where the only parameter is the number of pixels, *N*. Intensity-based methods, like Lucas-Kanade [57], Horn-Schunck [42], Hager-Belhumeur [39], require iterations to converge to a solution. Robustness is another superiority of phase-based methods to intensity-based ones. The Complex Wavelet Tracker proved itself immune to intensity perturbations in contrast to intensity-based methods.

Although the target is enclosed by a rectangular region, an inclusion of a mask that selects target pixels within the gate can easily be incorporated into the algorithm by adjusting the pixel weights. These weights are already incorporated into the solution.

The Complex Wavelet Tracker can be viewed as the phase-based alternative to the Lucas-Kanade's original image registration method [57] with the affine motion model [79]: Compatible accuracy with the addition of robustness.

## 7.2 Discussion on the Results

Tests started with an evaluation of the performance of the original CDWT based motion estimation method [59]. The results are compared to the Lucas-Kanade [57] and Horn-Schunck [40] motion estimation methods. It is observed that the CDWT based motion estimation is compatible in accuracy but is better in providing reasonable solutions even for difficult cases such as low texture and intensity changes.

Evaluations of the Complex Wavelet Tracker are performed in two ways. First, a set of *quantitative* tests are performed using controlled synthetic sequences in order to obtain quantitative results of the performance of the proposed method. The results are also compared with Lucas-Kanade's image registration method [57]. Second, *qualitative* tests are performed using real sequences acquired from different electro-optical imaging sources. The results are compared with Lucas-Kanade [57] and the Mean Shift Tracker [24].

The first group of quantitative tests aimed to explore the limits of estimation for affine deformations that the proposed algorithm can handle together with the accuracy of the estimations. Various tests have been performed to explore the limits of estimation for different motion types: translation, scale, shear, and rotation. The effect of the target size, the number of CDWT pyramid levels, and the motion model on the limits and accuracy of the algorithm are investigated. The algorithm showed acceptable limits for affine deformations that are suitable for tracking. The limits for each motion type were close to the theoretically computed ones. This justified our claim in using the CDWT as is (without modifications) to incorporate affine deformations to some extend that is acceptable for tracking purposes. The accuracy of the Complex Wavelet Tracker was also compatible with the accuracy of Lucas-Kanade [57].

The second group of tests aimed to explore the robustness of the algorithm to noise and intensity perturbations. Test data having different intensity patterns are used for the evaluations. The results are compared with Lucas-Kanade [57]. The proposed algorithm proved to be perfectly immune to intensity perturbations to a large extend in contrast to Lucas-Kanade [57] which, being intensity-based, experienced difficulties. For additive white Gaussian noise, a gradual increase in the estimation error is observed for both methods with increasing noise standard deviation.

The third group of tests aimed to investigate the robustness of the algorithm to different intensity patterns. Results obtained from previous tests are compared with respect to the three different test series: Lenna, Grass, and Cloud which represent different intensity patterns, or textures. The algorithm proved to be quite robust from this perspective. The results are also compared with Lucas-Kanade [57].

The last group of quantitative tests aimed to evaluate the long-term performance of the proposed algorithm under random tracking sequences. Random noise and random intensity perturbations are also added to the sequences. The Complex Wavelet Tracker proved to be quite accurate and robust for these test in comparison to Lucas-Kanade.

After the quantitative tests, qualitative evaluations have been performed on various real sequences. The results are compared with Lucas-Kanade [57] and the Mean Shift Tracker [24].

For the air sequences, the aircrafts were very dynamic targets resulting in a diverse set of deformations. The Complex Wavelet Tracker was very successful in updating the track gate according to the changes of the target and maintained track until the last frame for most of the sequences. The smoothness of the background actually favored the tracker for this kind of sequences. The Lucas-Kanade, however, experienced difficulties when there remained little contrast between the target and backgroung and during changes in the brightness of the scene.

In the Pursaklar sequences, the targets were in front of a cluttered background. Both color and infrared versions of the same scene are used to observe also the change in the performance for different imaging sources. Compared to the color counterpart, the infrared sequence was smoother and lower in contrast. Small drifts are observed from the actual target, especially the scaling was less than the actual one, however, the gate was maintained on the target. For the color sequence, the tracking was more confident, with a small drift in the track gate.

For all the real sequences, Lucas-Kanade performed similar to the Complex Wavelet Tracker. However, for half of the sequences, it lost the target before the last frame is reached. The Mean Shift Tracker maintained track for all sequences, but, it experienced difficulties in maintaining the gate continuously on the target and in updating the size of the gate appropriately.

The computational complexity of the proposed algorithm is $O(N)$ and can be implemented in real-time. Its computational time is larger than the Mean Shift

Tracker, but usually smaller than Lucas-Kanade depending on the number of iterations performed.

On the overall, the proposed tracker proved to be a robust and accurate tracking algorithm suitable to track regions of a wide variety. It can be implemented in real-time. It has its advantages of being robust to illumination changes in contrast to intensity-based methods and can handle a wider range of targets with compatible accuracy.

## 7.3 Suggestions for Further Work

Further work to improve the proposed Complex Wavelet Tracker can be conducted on the items captured in Chapter 5, especially for the use and adaptation of the algorithm to other application areas. In addition to this, further work can also be directed to the development of the Complex Wavelet Tracking Framework introduced in Chapter 5.

Work can be directed to develop a more adequate quality measure to better reflect the confidence of the track parameter estimates. This measure could be directed to aid for both confidence and occlusion detection. An initial derivation of a confidence measure is given in Section 5.1.2.

More work can be conducted on using a masking kernel to improve the tracking performance. A 2-D Gaussian or Epanechnikov kernel [23] can be used in order to give more weight to the center coefficients so that the estimations are least affected by background pixels that enter the track gate. The mask can also be updated to match the target shape if it can be extracted from the background by using a kind of detection or segmentation algorithm. Then, only the target pixels would be taken into account for the estimation of the track parameters.

Work can be conducted in fusing the algorithm with an appearance- or intensity-based technique like Lucas-Kanade [57]. The difficulty in the fusion process lies in the merging strategy. This fusion would combine the accuracy of the Lucas-Kanade with the robustness of the Complex Wavelet Tracker to obtain a superior method.

As the scope of this work was to establish an algorithmic base, no considerations have been made on improving the tracking process. So, work could be conducted on this area also. One necessity is to use a parameter filter in order to cancel out unreliable or out of limit estimates and replace those with an estimate based on the previous estimates. This filter might not be as complex as a Kalman filter [45] but should be relatively simple to allow fast and random changes of parameters through successive frames.

## 7.4 Conclusion

A new method is presented for the parametric tracking of areas based on the Complex Discrete Wavelet Transform (CDWT). The method is a generalization of the CDWT based motion estimation algorithm [60] developed by Magarey and Kingsbury. It incorporates a parametric motion model and estimates the motion of regions instead of individual pixels. In this way, with an increase in the support, robustness and accuracy of the estimation is increased as well.

Numerous simulations have been performed, including both *quantitative* and *qualitative* tests, to evaluate the performance of the proposed method. Quantitative tests are performed using synthetic test sequences and the estimated results are compared with true data. The practical limits of deformation that the Complex Wavelet Tracker can handle are explored using tests for each motion type. It is observed that the results are close to theoretically computed ones and are sufficient for tracking. By increasing the CDWT levels used, the practical limits can also be increased. Robustness tests have been performed for noise and intensity perturbations. Results have been compared to Lucas-Kanade [57]. It is observed that, in contrast to intensity-based methods, the proposed method shows perfect robustness to illumination changes.

Qualitative tests have been performed on real sequences. The results have been compared with Lucas-Kanade [57] and the Mean Shift Tracker [23]. It is observed that the tracker is quite successful for tracking a wide range of targets in a wide range of applications. It is compatible in accuracy to Lucas-Kanade [57] and is robust in terms of intensity perturbations. Hence, the proposed method proves itself to be quite appropriate for tracking a wide range of targets.

As this is only an algorithmic base, and nothing special is done to improve the tracking performance, further work can be directed to this aspect. Despite this fact, the proposed tracker showed superior tracking performance on various kinds of situations.

As a result, we can conclude that the proposed method can be used quite successfully for area tracking. It is a robust tracker that can handle affine deformations and can be executed in real-time. It has provides compatible accuracy to its intensity-based counterparts with the addition of robustness to illumination changes. It can also handle more difficult cases like smooth, low textured regions.

# REFERENCES

[1] H. F. Ates and M. T. Orchard. "Block Motion Estimation Using Wavelet Filtering," *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, pp. 2079-2082, 2000.

[2] G. V. d. Auwera, G. L. Munteanu, and J. Cornelis, "Video Coding based on Motion Estimation in the Wavelet Detail Images," *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, pp. 2801-2804, 1998.

[3] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework: Part 1," CMU-RI-TR-02-16, *Robotics Institute*, *Carnegie Mellon University*, 2003.

[4] S. Baker and I. Matthews implementation source code of Lucas-Kanade affine motion estimation algorithm. Web site: http://www.cs.cmu.edu/~iainm/lk20p1.tgz and last access date: 26 June 2007.

[5] J. L. Barron, D. J. Fleet and S. S. Beauchemin, "Performance of Optical Flow Techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43-77, 1994.

[6] J. L. Barron implementation source code of Lucas-Kanade and Horn-Schunck optical flow estimation algorithms. Web site: ftp://ftp.csd.uwo.ca/pub/vision/ and last access date: 24 Jun 2007.

[7] A. Behrad, S. A. Motamedi, and K. Madani, "A New Algorithm for Target Tracking using Fuzzy-Edge-based Feature Matching and Robust Statistics," *IEEE International Conference on Image Processing*, pp. 577-580, 2002.

[8] C. P. Bernard, "Discrete Wavelet Analysis for Fast Optic Flow Computation," *Centre de Mathématiques Appliquées*, International Report RI415, École Polytechnique, 1999.

[9] C. P. Bernard, "Fast Optic Flow Computation with Discrete Wavelets," *Centre de Mathématiques Appliquées*, International Report RI365, École Polytechnique, 1997.

[10] J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker," *Intel Corp., Microprocessor Research Labs,* Technical Report, 2000.

[11] R. N. Bracewell, "Affine Theorem for Two-Dimensional Fourier Transform," *Electronics Letters*, vol. 29, no. 3, Feb. 1993.

[12] F. Bremond and M. Thonnat, "Tracking Multiple Nonrgid Objects in Video Sequences," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 585-591, 1998.

[13] E. Brookner, *Tracking and Kalman Filtering Made Easy*, John Wiley & Sons Inc., 1998.

[14] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods," *International Journal of Computer Vision*, vol. 61, no. 3, pp. 211-231, 2005.

[15] C. E. Caefer, J. Silverman, and J. M. Mooney, "Optimization of Point Target Tracking Filters", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 1, pp. 15-25, 2000.

[16] R. Canals, A. Roussel, J.-L. Famechon, and S. Treuillet, "A Biprocessor-Oriented Vision-Based Target Tracking System," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 2, pp. 500-506, 2002.

[17] W. Cai, *High Performance Shift Invariant Motion Estimation and Compensation in Wavelet Domain Video Compression*, Ph. D. Thesis, Florida International University, 2003.

[18] W. Cai and M. Adjouadi, "An Efficient Approach of Fast Motion Estimation and Compensation in Wavelet Domain Video Compression," *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, pp. 977-980, 2004.

[19] G. Castellano, *Investigation and Application of a Complex Wavelet Transform Algorithm for the Estimation of Optical Flow*, Ph.D. Thesis, King's College, University of London, June 1999.

[20] I. Cohen and G. Medioni, "Detecting and Tracking Moving Objects for Video Surveillance," *IEEE Proceedings on Computer Vision and Pattern Recognition*, 1999.

[21] I. Cohen and G. Medioni, "Detecting and Tracking Moving Objects in Video from an Airborne Observer," *DARPA Image Understanding Workshop*, 1998.

[22] I. Cohen and G. Medioni, “Detection and Tracking of Objects in Airborne Video Imagery,” *IEEE Computer Vision and Pattern Recognition: Interpretation of Visual Motion Workshop*, 1998.

[23] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-Based Object Tracking,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, May 2003.

[24] D. Comaniciu, *Nonparametric Robust Methods for Computer Vision*, Ph. D. Thesis, The State University of New Jersey, 2000.

[25] S. Cui, *Motion Estimation and Compensation in the Redundant Wavelet Domain*, Ph. D. Thesis, Computer Engineering, Mississippi State University, 2003.

[26] S. Cui, Y. Wang, and J. E. Fowler, “Mesh-Based Motion Estimation and Compensation in the Wavelet Domain using a Redundant Transform,” *Proceedings of the IEEE Conference on Image Processing*, vol. 1, pp. 693-696, 2002.

[27] D. Davies, P. Palmer, and M. Mirmehdi, “Detection and Tracking of Very Small Low Contrast Objects”, *Proceedings of the 9th British Machine Vision Conference*, Sept. 1998.

[28] R. DeVore, A. Petukhov, and R. Sharpley, “Motion Estimation with the Redundant Wavelet Transform,” *IEEE International Workshop on Digital and Computational Video*, pp. 53-59, 2002.

[29] D. J. Fleet, *Measurement of Image Velocity*, Ph.D. Thesis, Department of Computer Science, University of Toronto, 1991.

[30] D. J. Fleet and A. D. Jepson, "Computation of Component Image Velocity from Local Phase Information," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 77-104, 1990.

[31] F. C. A. Fernandes, R. L. C. v. Spaendonck and C. S. Burrus, “A New Framework for Complex Wavelet Transforms,” *IEEE Transactions on Signal Processing*, pp. 1825-1837, July 2003.

[32] F. C. A. Fernandes, R. L. C. v. Spaendonck, and C. S. Burrus, “A New Directional, Low-Redundancy, Complex-Wavelet Transform,” *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, pp. 3653-3656, 2001.

[33] G. L. Foresti, “Object Recognition and Tracking for Remote Video Surveillance,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 7, pp. 1045-1062, 1999.

[34] J. M. Fitts, “Video Correlation Tracker,” *United States Patent*, no. 4.133.004, Jan 1979.

[35] M. F. Fu, O. C. Au, and W. C. Chan, “Low Band Shift (LBS) Motion Estimation with Symmetric Padding in Wavelet Domain,” pp. 13-16, 2002.

[36] C. A. Gentile, *Robust Tracking with Parts in the Presence of Severe Occlusion*, Ph. D. Thesis, The Pennsylvania State University, 2001.

[37] O. G. Guleryuz, “Maximizing Uniform Translational Motion - Motion Estimation with the Haar Transform and Dynamic Programming,” *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, 2000.

[38] O. G. Guleryuz and L. Sendur, “Globally Optimal Wavelet-Based Motion Estimation using Interscale Edge and Occlusion Models,” *Proceedings of the SPIE*, vol. 5308, pp. 428-439, 2004.

[39] G. D. Hager and P. N. Belhumeur, “Efficient Region Tracking With Parametric Models of Geometry and Illumination,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025-1039, Oct 1998.

[40] C. He, Y. F. Zheng, and S. C. Ahalt, “Object Tracking Using the Gabor Wavelet Transform and the Golden Section Algorithm,” *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 528-538, 2002.

[41] B. K. P. Horn, *Computer Vision*, MIT Press, Cambridge, Mass., 1986.

[42] B. K. P. Horn and B. G. Schunck, “Determining Optical Flow,” *Artificial Intelligence*, vol. 17, pp. 185-204, 1981.

[43] S. W. Hwang, E. Y. Kim, S. H. Park, and H. J. Kim, “Object Extraction and Tracking using Genetic Algorithms,” *IEEE International Conference on Image Processing*, vol. 2, pp. 383-386, 2001.

[44] M. Isard and A. Blake, “CONDENSATION - Conditional Density Propagation for Visual Tracking,” *International Journal of Computer Vision (IJCV)*, vol. 1, no. 29, pp. 5-28, 1998.

[45] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35-45, 1960.

[46] N. Kingsbury, “Complex Wavelets for Shift Invariant Analysis and Filtering of Signals,” *Journal of Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234-253, 2001.

[47] N. Kingsbury and J. Magarey, "Wavelet Transforms in Image Processing," *Proceedings of the European Conference on Signal Analysis and Processing*, pp. 24-27, 1997.

[48] N. Kingsbury and J. Magarey, "Wavelets in Image Analysis: Motion and Displacement Estimation," *Proceedings of the Irish Digital Signal Processing and Control Conference*, pp. 199-217, 1996.

[49] D. Koller, J. Weber, and J. Malik, "Robust Multiple Car Tracking with Occlusion Reasoning," *Proceedings of the 3rd European Conference on Computer Vision*, Stockholm, vol. 1, pp. 189-196, 1994.

[50] J. Konrad and E. Dubois, "Estimation of Image Motion Fields: Bayesian Formulation and Stochastic Solution," *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, 1988, pp. 1072-1075.

[51] S. Kumar, M. Biswas, and T. Q. Nguyen, "Global Motion Estimation in Frequency and Spatial Domain," *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASPP)*, vol. 3, pp. 333-336, May 2004.

[52] P. A. Laplante and A. D. Stoyenko, *Real-Time Imaging: Theory, Techniques, and Applications*, IEEE Press, 1996.

[53] X. Li, "Fast and Efficient Block Motion Estimation in the Wavelet Space," *IEEE International Conference on Information Technology: Computers and Communications (ITCC)*, 2003.

[54] J. Lim, H. K. Cho, and J. B. Ra, "An Improved Video Object Tracking Algorithm based on Motion Re-Estimation," *IEEE International Conference on Image Processing*, pp. 339-342, 2000.

[55] J. Lim and J. B. Ra, "A Semantic Video Object Tracking Algorithm using Three-Step Boundary Refinement," *IEEE International Conference on Image Processing*, vol. 2, pp. 159-163, 1999.

[56] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, "Moving Target Classification and Tracking from Real-Time Video," *IEEE Workshop on Applications of Computer Vision*, pp. 8-14, 1998.

[57] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proceedings of DARPA Image Understanding Workshop*, pp. 121-130, 1981.

[58] L. Lucchese, S. Leorin, and G. M. Cortelazzo, "Estimation of Two-Dimensional Affine Transformations Through Polar Curve Matching and Its

Application to Image Mosaicking and Remote Sensing Data Registration," *IEEE Transactions on Image Processing*, vol. 15, no. 10, Oct. 2006.

[59] J. Magarey, *Motion Estimation using Complex Wavelets*, Ph.D. Thesis, Department of Engineering, University of Cambridge, February 1997.

[60] J. Magarey and N. Kingsbury, "Motion Estimation Using a Complex-Valued Wavelet Transform," *IEEE Transactions on Signal Processing*, vol. 46, no. 4, pp. 1069-1084, 1998.

[61] J. Magarey and N. Kingsbury, "Motion Estimation Using Complex Wavelets," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. 2371-2374, 1996.

[62] J. Magarey and N. Kingsbury, "An Improved Motion Estimation Algorithm using Complex Wavelets," *IEEE Proceedings of the International Conference on Image Processing*, pp. 969-972, 1996.

[63] J. Magarey and N. Kingsbury, "Motion Estimation Using a Complex-Valued Wavelet Transform," *Proceedings of the SPIE: Wavelet Applications in Signal and Image Processing*, vol. 2825, pp. 674-685, 1996.

[64] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674-693, 1989.

[65] I. Matthews, T. Ishikawa, and S. Baker, "The Template Update Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810-815, June 2004.

[66] F. A. Mujica, J.-P. Leduc, R. Murenzi and M. J. T. Smith, "A New Motion Parameter Estimation Algorithm Based on the Continuous Wavelet Transform," *IEEE Transactions on Image Processing* vol. 9, no. 5, pp. 873-888, 2000.

[67] S. C. Nassif, *Cooperative Windowing for Real-Time Visual Tracking*, Ph.D. Thesis, McMaster University, 1997.

[68] K. M. Nickels, *Model-Based Tracking of Articulated Objects*, Ph.D. Thesis, University of Illinois at Urbana-Champaign, 1998.

[69] K. Nummiaro, E. Koller-Meier and L. V. Gool, "Object Tracking with an Adaptive Color-Based Particle Filter." *Symposium for Pattern Recognition of the DAGM*, 2002.

[70] H.-W. Park and H.-S. Kim, "Motion Estimation Using Low-Band-Shift Method for Wavelet-Based Moving-Picture Coding," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 577-587, 2000.

[71] E. Polat, *Tracking Multiple Objects using Multiple Hypothesis Tracking Framework*, Ph.D. Thesis, The Pennsylvania State University, 2002.

[72] F. M. Porikli, *Video Object Segmentation*, Ph.D. Thesis, Polytechnic University, 2002.

[73] C. E. Rasmussen, *Integrating Multiple Visual Cues for Robust Tracking*, Ph.D. Thesis, Yale University, 2000.

[74] T. K. Robb, *A Comparison of Conventional and Neural Network Data Assocation Techniques for Multi-Target Tracking*, Ph.D. Thesis, Royal Military College of Canada, 1999.

[75] T. Schoepflin, V. Chalana, D. R. Haynor and Y. Kim, "Video Object Tracking with a Sequential Hierarchy of Template Deformations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 11, pp. 1171-1182, 2001.

[76] B. J. Scholla, Z. W. Pylyshynb, and J. Feldman, "What is a visual object - Evidence from target merging in multiple object tracking," *Cognition*, vol. 80, no. 1-2, pp. 159-177, 2001.

[77] N. Sebe, C. Lamba, and M.S. Lew, "An Overcomplete Discrete Wavelet Transform for Video Compression," *ICME*, 2002.

[78] K. Seyrafi and S. A. Hovanessian, *Introduction to Electro-Optical Imaging and Tracking Systems*, Artech House Inc., 1993.

[79] J. Shi and C. Tomasi, "Good Features to Track," *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, June 1994.

[80] E. P. Simoncelli, W. T. Freeman, E. H. Adelson, and D. J. Heeger, "Shiftable Multi-Scale Transforms," *IEEE Transactions on Information Theory*, vol. 38, no.2, pp. 587-607, March 1992.

[81] A. M. Tekalp, *Digital Video Processing*, Prentice-Hall Inc., 1995.

[82] M. Y. Tjoa, Y. Imanishi, N. Yamane, and Y. Morikawa, "A Motion Compensation Method using Least Squares Motion Estimation Filter in Wavelet Domain," *IEEE Proceedings of the Joint Conference of the International Conference on Information, Communications and Signal Processing and Pacific-Rim Conference on Multimedia (ICICS-PCM)*, 2003.

[83] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, "Making Good Features Track Better," *Proceedings IEEE Computer Vision and Pattern Recognition*, pp.1998.

[84] K. Toyama, *Robust Vision-Based Object Tracking*, Ph.D. Thesis, Yale University, 1998.

[85] S. A. Vigus, D. R. Bull, and C. N. Canagarajah, “Video Object Tracking using Region Split and Merge and a Kalman Filter Tracking Algorithm,” *IEEE International Conference on Image Processing*, 2001.

[86] S. Wong, “Advanced Correlation Tracking of Objects in Cluttered Imagery,” *Proceedings of the SPIE: Acquisition, Tracking, and Pointing XIX*, Orlando, USA, 2005.

[87] Y.-T. Wu, *Image Registration using Wavelet-based Motion Model and its Applications*, Ph. D. Thesis, Department of Electrical Engineering, University of Pittsburgh, 1997.

[88] Y.-T. Wu, T. Kanade, J. Cohn and C.-C. Li, “Optical Flow Estimation Using Wavelet Motion Model,” *International Conference on Computer Vision (ICCV)*, 1998.

[89] H. Xiong, T. Zhang and Y. S. Moon, “A Translation- and Scale-Invariant Adaptive Wavelet Transform," *IEEE Transactions on Image Processing*, vol. 9, no. 12, pp. 2100-2108, 2000.

[90] Y. Yuan and M. K. Mandal, “Low-Band-Shifted Hierarchical Backward Motion Estimation and Compensation for Wavelet-Based Video Coding,” *Proceedings of ICVGIP*, 2002.

# APPENDIX A

# MOTION ESTIMATION RESULTS

This Appendix presents the results of the motion estimation simulations of the CDWT-ME algorithm of Magarey and Kingsbury in comparison to Lucas-Kanade and Horn-Schunck motion estimation algorithms. The error measures used for the results are presented in Section 6.1.1. A discussion on the results is given in Section 6.1.2. The Appendix starts with an introduction of the test sequences followed by a presentation of the results.

## A.1 Test Sequences for Motion Estimation

For the comparison of the algorithms we used the test sequences Barron *et al* has used in [5] to compare the performances of several optical flow methods. There are seven synthetic sequences with known correct flows and four real sequences. The synthetic image sequences are: Sine-B, Sine-C, Square-1, Square-2, Yosemite Fly-Through, Translating Tree and Diverging Tree; the real sequences are: NASA, Hamburg Taxi, Rubik and Trees.

### A.1.1 Synthetic Image Sequences

There are seven synthetic image sequences used by Barron *et al* in [5]:

**Sinusoid-1:** This sequence is created by superimposing two sinusoids moving with speeds 1.63 at 54° and 1.02 at -27° for a perceived velocity of (1.585, 0.863) pixels/frame. The spatial wavelength is 6 pixels per cycle. This sequence with its corresponding correct flow is shown in Figure A.1.

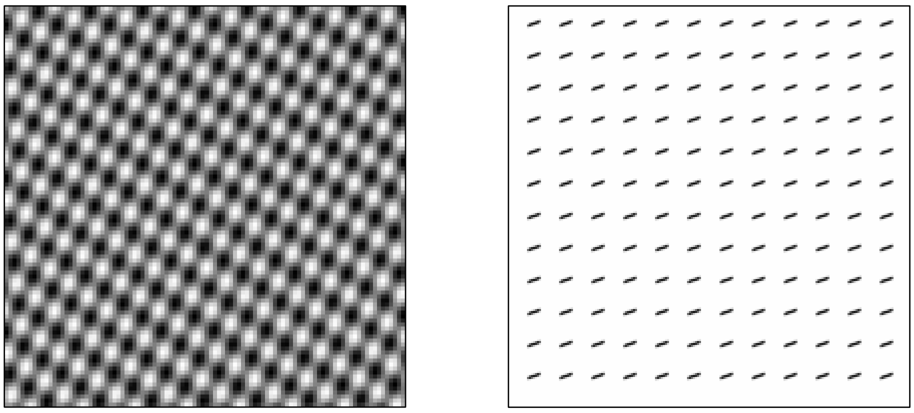


**Figure A.1** The Sinusoid-1 Sequence with its correct flow field.

**Sinusoid-2:** This sequence is created by superimposing two sinusoids both moving with speeds of 1.0 pixel/frame at 0° and 90° orientation for a perceived velocity of (1.0, 1.0) pixels/frame. The spatial wavelength is 16 pixels per cycle. This sequence with its corresponding correct flow is shown in Figure A.2.

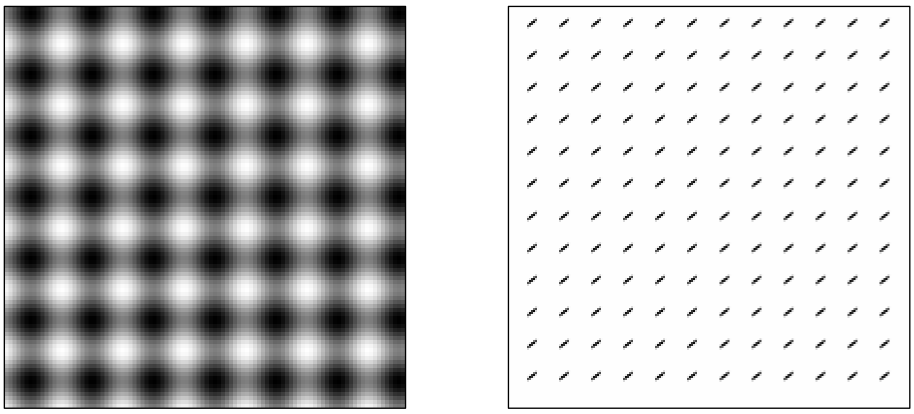


**Figure A.2** The Sinusoid-2 Sequence with its correct flow field.

**Translating Square-1:** In this sequence, a white square is moving with velocity (1.0, 1.0) pixels/frame on a black background. This sequence with its corresponding correct flow is shown in Figure A.3.

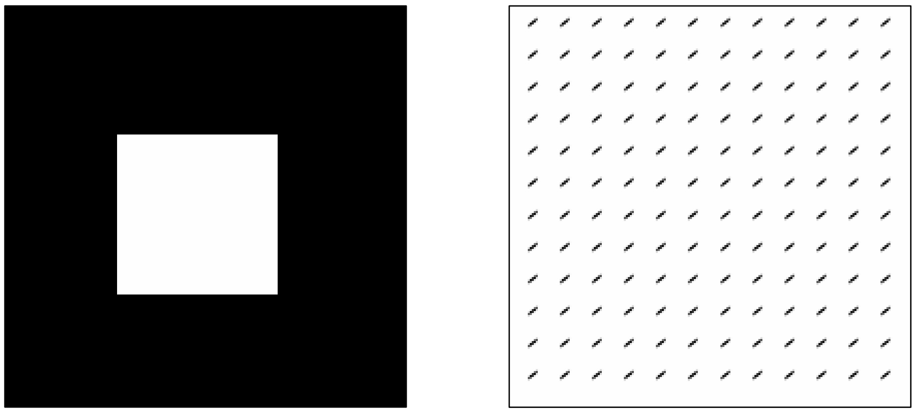


**Figure A.3** The Translating Square-1 Sequence with its correct flow field.

**Translating Square-2:** In this sequence, a blurred black square is moving with velocity (4/3, 4/3) on a white background. This sequence with its corresponding correct flow is shown in Figure A.4.

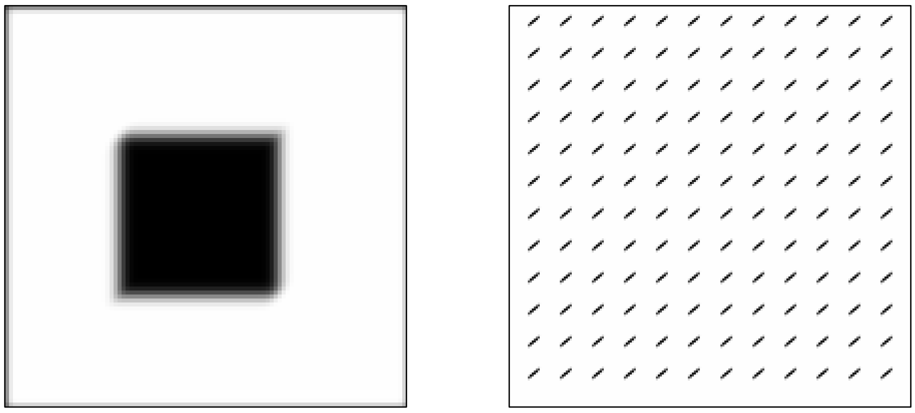


**Figure A.4** The Translating Square-2 Sequence.

**Translating Tree:** The translating Tree sequence is produced by David Fleet. The sequence is created by moving a synthetic camera relative to a planar image of a tree. The velocity ranges from (1.73, 0.0) pixels/frame on the left to (2.3, 0.0) pixels/frame on the right. The sequence with its corresponding correct flow is shown in Figure A.5.

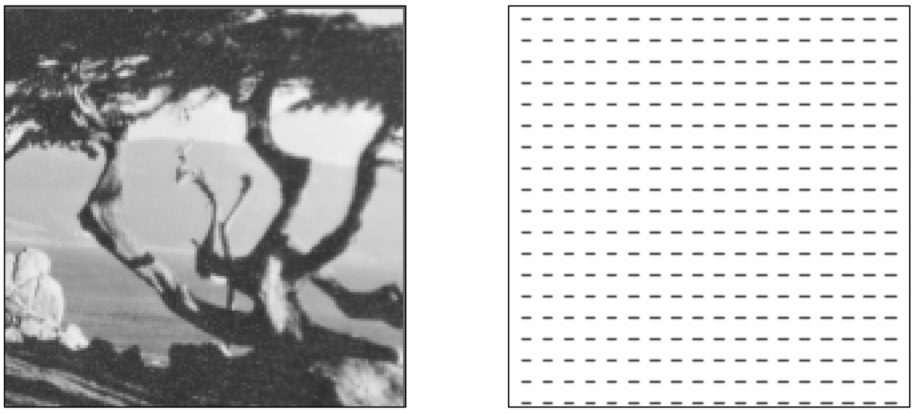


**Figure A.5** The Translating Tree Sequence.

**Diverging Tree:** The Diverging Tree sequence is also produced by David Fleet. The images are created by moving a synthetic camera relative to a planar image of a tree. The velocity speeds range from 0 in the middle (at the FOE) to 1.4 pixels/frame on the left and 2.0 pixels/frame on the right. The sequence with its corresponding correct flow is shown in Figure A.6.

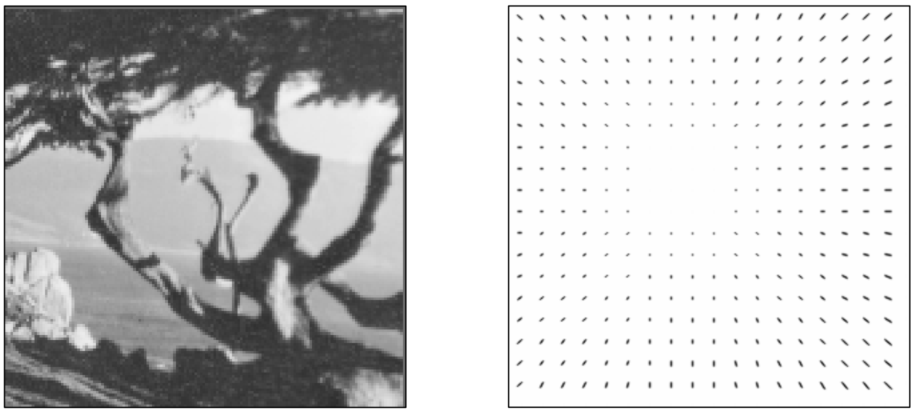


**Figure A.6** The Diverging Tree Sequence.

**Yosemite Sequence:** The Yosemite Fly-Through sequence is produced by Lynn Quam. The motion of the clouds is 2 pixels to the right while the rest of the flow is divergent, with speed of about 5 pixels/frame in the lower left corner. The sequence with its corresponding correct flow is shown in Figure A.7.

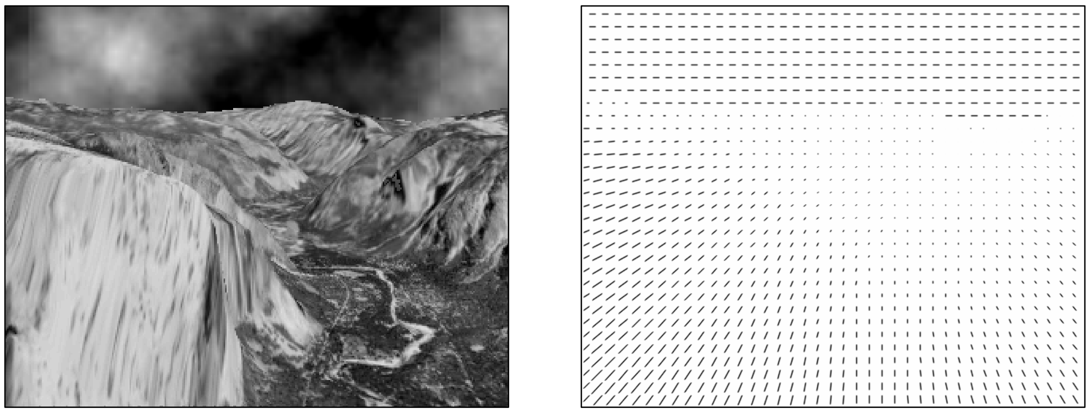


**Figure A.7** The Yosemite Sequence.

### A.1.2 Real Image Sequences

There are four real image sequences used by Barron *et al*:

**Rotating Rubik Cube:** A rotating Rubik cube on a microwave turntable, produced by Richard Szeliski. The motion field induced by the rotation of the cube includes velocities less than 2 pixels/frame. The velocities on the turntable range from 1.2 to 1.4 pixels/frame, and those on the cube are between 0.2 and 0.5 pixels/frame.

**Hamburg Taxi Sequence:** The Hamburg taxi scene is from the University of Hamburg. There are four moving objects: a taxi turning the corner, a car in the lower left, driving from left to right, a van in the lower right driving from right to left and a pedestrian in the upper right. Image speeds of the four moving objects are approximately 1.0, 3.0, 3.0, and 0.3 pixels/frame, respectively.

**NASA Sequence:** A subimage of the original NASA coke can sequence from Nasa-Ames. This is a purely diverging sequence; the camera moves along its line of sight toward the coke can near the center of the image. Image velocities are typically less than 1 pixel/frame.

**SRI Sequence:** The SRI trees sequence; the motion is translation in the fronto-parallel plane. The camera translates parallel to the ground plane, perpendicular to its line of sight, in front of clusters of trees. Velocities are as large as 2 pixels/frame.

These sequences are shown in Figure A.8.

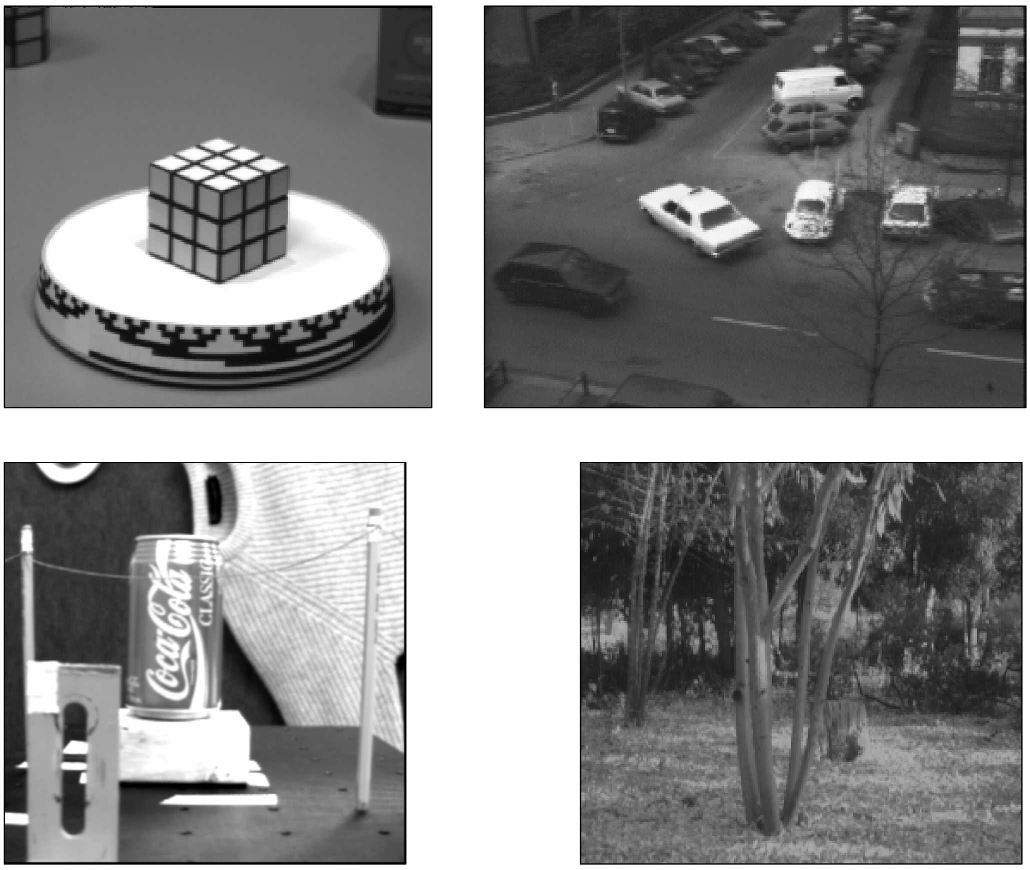


**Figure A.8** The Real Sequences. The Rotating Rubik Cube (upper left), the Hamburg Taxi Sequence (upper right), the NASA Sequence (lower left), and the SRI Sequence (lower right).

## A.2 Motion Estimation Performance of Magarey's CDWT-ME

The flow images have been displayed with flow vectors scaled by two for every 8th vector. The angle error is superimposed as a background image with white indicating no error and as the color gets darker, the error increases. A red dot represents a pixel where no flow is computed or is thresholded out. The thresholded out regions are also shown with a yellow background.

Regarding the results given in the tables, for the CDWT-ME algorithm, although it produces results with 100% density, we excluded a four pixel boundary in computing the error statistics to eliminate boundary errors.

For the Lucas-Kanade [57] and Horn-Schunck [40] algorithms, we used the same parameters and smoothing values as Barron *et al* has used in his comparison in [5]. In addition to this, we included also non-thresholded values and flow fields as a comparison of the whole flow field. For a point-to-point comparison, we also gave the errors for those points that are thresholded out using Lucas-Kanade confidence values. These results are given in the last row of each table.

In the tables, *Tau* shows the threshold for Lucas-Kanade and Horn-Schunck algorithms. For *Tau* = 0.0, no thresholding is applied and the errors are computed for the whole image.

For the rest of this section, we will refer to the Lucas-Kanade [57] algorithm as "LK", and to the Horn-Schunck [40] algorithm as "HS".

### A.2.1 The Sinusoid-1 Sequence

The results for the first sinusoid are shown in Figure A.9 and in Table A.1. The performance of the CDWT-ME for the sinusoid was worse than the other two algorithms. Although the mean angle error is close to the HS's error, examining the flow field of CDWT-ME reveals a non-uniform flow field. The LK has produced a quite accurate flow field.

**Table A.1** Results of the Sinusoid-1 Sequence.

| ***Algorithm*** | ***Density*** | | ***Angle Error*** | | ***Magnitude Error*** | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *Std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 8464 | 84.6 | 4.8666 | 0.9700 | 0.2354 | 0.0498 |
| CDWT-ME (Conf. Thresh. = 0.95) | 8464 | 84.6 | 4.8666 | 0.9700 | 0.2354 | 0.0498 |
| Lucas-Kanade (Tau = 1.0) | 8464 | 84.6 | 2.4697 | 0.1244 | 0.1109 | 0.0047 |
| Horn-Schunck (Tau = 0.0) | 8464 | 84.6 | 4.2531 | 0.4195 | 0.2005 | 0.0196 |
| CDWT-ME (at LK Tau = 1.0) | 8464 | 84.6 | 4.8666 | 0.9700 | 0.2354 | 0.0498 |

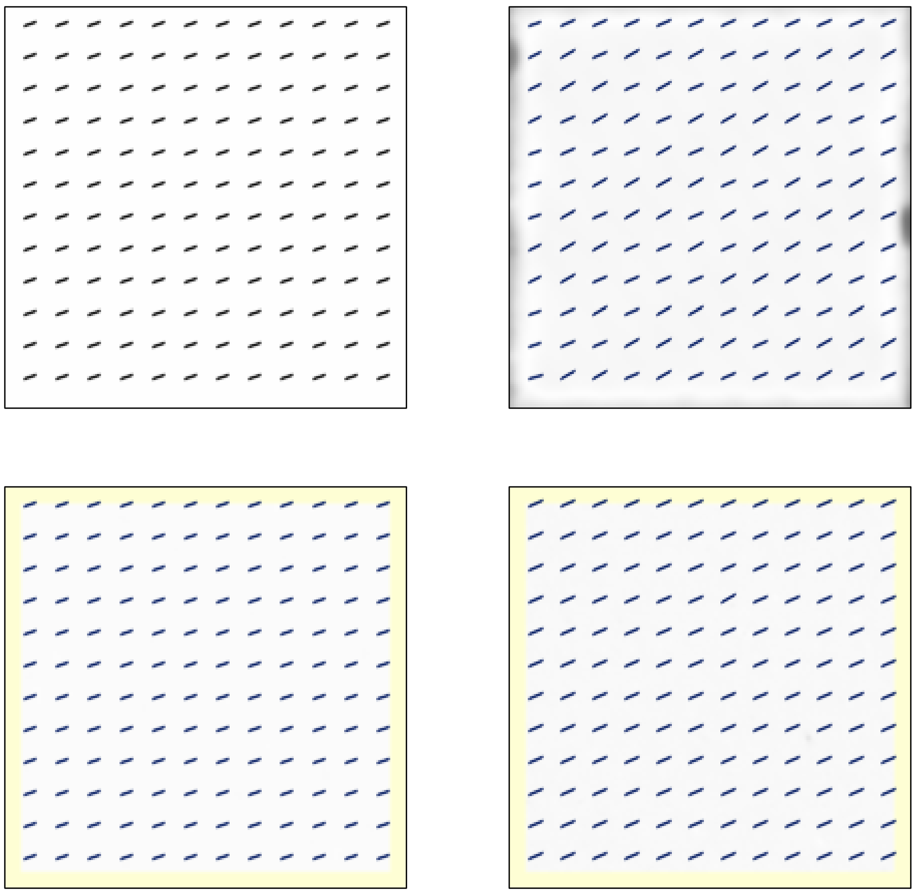


**Figure A.9** Results of the Sinusoid-1 Sequence: The correct flow (upper left), the CDWT-ME (upper right), Lucas-Kanade (lower left), and Horn-Schunck (lower right).

## A.2.2 The Sinusoid-2 Sequence

The results for the second sinusoid are shown in Figure A.10 and in Table A.2. The CDWT-ME produced a similar result as in the first sinusoid. The LK and HS have produced a nearly perfect flow field.

**Table A.2** Results of the Sinusoid-2 Sequence.

| *Algorithm* | *Density* | | *Angle Error* | | *Magnitude Error* | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 8464 | 84.6 | 4.2247 | 2.2299 | 0.1660 | 0.0610 |
| CDWT-ME (Conf. Thresh. = 0.95) | 8464 | 84.6 | 4.2247 | 2.2299 | 0.1660 | 0.0610 |
| Lucas-Kanade (Tau = 1.0) | 8464 | 84.6 | 0.0001 | 0.0027 | 0.0000 | 0.0001 |
| Horn-Schunck (Tau = 0.0) | 8464 | 84.6 | 0.0077 | 0.0170 | 0.0004 | 0.0009 |
| CDWT-ME (at LK Tau = 1.0) | 8464 | 84.6 | 4.2247 | 2.2299 | 0.1660 | 0.0610 |

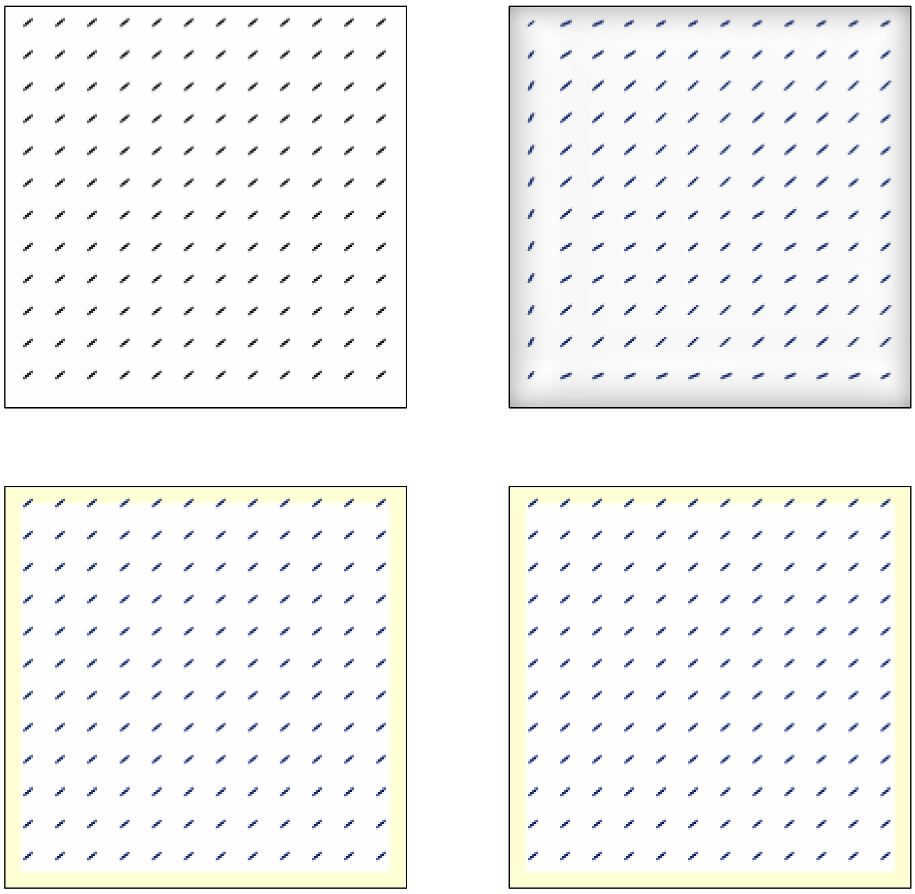


**Figure A.10** Results of the Sinusoid-2 Sequence: The correct flow (upper left), the CDWT ME (upper right), Lucas-Kanade (lower left), and Horn-Schunck (lower right).

### A.2.3 The Translating Square-1 Sequence

In this sequence, since the square and the background have no texture and are in unique color level, flow has been produced only around the edges of the square. By examining the flow fields, the flow estimates of CDWT-ME are highly more reasonable than the ones produced by HS. The overall errors also support this observation. The mean error values are given in Table A.3. The flow fields are shown in Figure A.11.

**Table A.3** Results of the Translating Square-1 Sequence.

| *Algorithm* | *Density* | | *Angle Error* | | *Magnitude Error* | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 960 | 9.6 | 8.3020 | 4.0317 | 0.1780 | 0.0777 |
| CDWT-ME (Conf. Thresh. = 0.95) | 944 | 9.4 | 8.4205 | 3.9602 | 0.1805 | 0.0759 |
| Horn-Schunck (Tau = 1.0) | 1836 | 18.4 | 24.2593 | 9.2389 | 0.3721 | 0.1368 |

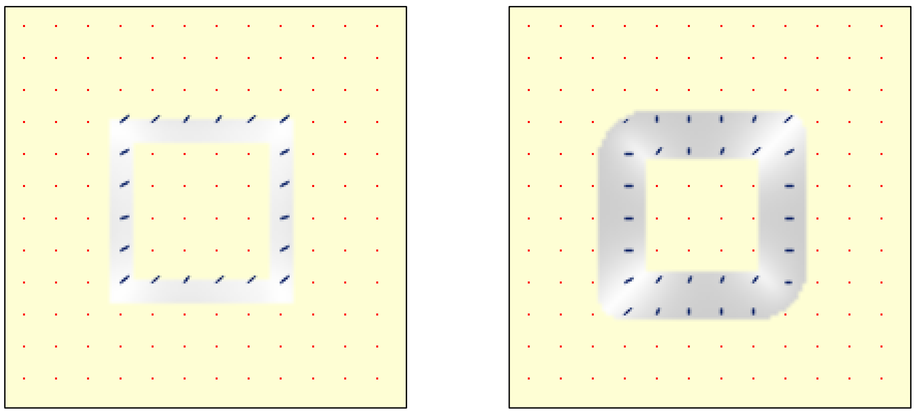


**Figure A.11** Results of the Translating Square-1 Sequence: The CDWT-ME (left) and Horn-Schunck (right).

### A.2.4 The Translating Square-2 Sequence

In this sequence, the square is blurred, but both the background and square are still in unique color, except for the edges. The blurring increased the density for both methods, however, the resulting errors have increased. Since motion in smooth texture is more difficult to estimate. The flow computed by the CDWT-ME has produced a more reasonable flow field with lower errors than the HS algorithm. The flow images are shown in Figure A.12 and the corresponding results are given in Table A.4.

**Table A.4** Results of the Translating Square-2 Sequence.

| *Algorithm* | *Density* | | *Angle Error* | | *Magnitude Error* | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 2528 | 25.3 | 17.9693 | 17.8374 | 0.6569 | 0.4951 |
| CDWT-ME (Conf. Thresh. = 0.95) | 2424 | 24.2 | 18.3750 | 18.0898 | 0.6642 | 0.5030 |
| Horn-Schunck (Tau = 1.0) | 2282 | 22.8 | 26.4638 | 10.8562 | 0.5191 | 0.2426 |

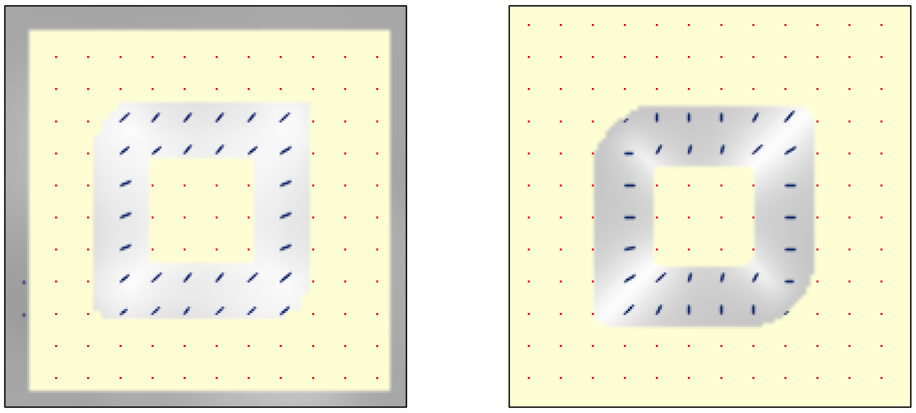


**Figure A.12** Results of the Translating Square-2 Sequence: The CDWT-ME (left) and Horn-Schunck (right).

### A.2.5 The Translating Tree Sequence

For the translating tree sequence, the LK has produced a quite accurate flow field. The performance of CDWT-ME was slightly better than the HS for the thresholded values. The gradient computations were quite accurate for this sequence whose only motion is the uniform translation of the whole image. Since the performance of the LK and HS algorithms highly depend upon the gradient computation, this sequence favored these methods. For the CDWT-ME, the errors were mostly near the borders of the image. The flow images are shown in Figure A.13 and the corresponding results are given in Table A.5.

**Table A.5** Results of the Translating Tree Sequence.

| *Algorithm* | *Density* | | *Angle Error* | | *Magnitude Error* | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 20108 | 89.4 | 2.4879 | 4.3515 | 0.0966 | 0.1385 |
| CDWT-ME (Conf. Thresh. = 0.95) | 17528 | 77.9 | 1.8694 | 2.2987 | 0.0790 | 0.0984 |
| Lucas-Kanade (Tau = 1.0) | 6845 | 30.4 | 0.6486 | 0.6592 | 0.0272 | 0.0276 |
| Lucas-Kanade (Tau = 0.0) | 14884 | 66.2 | 1.0151 | 1.2552 | 0.0471 | 0.0575 |
| Horn-Schunck (Tau = 5.0) | 7913 | 35.2 | 1.8865 | 2.4110 | 0.0876 | 0.1396 |
| Horn-Schunck (Tau = 0.0) | 14884 | 66.2 | 2.0185 | 2.2715 | 0.0924 | 0.1305 |
| CDWT-ME (at LK Tau = 1.0) | 6841 | 30.4 | 1.9263 | 1.8257 | 0.0698 | 0.0659 |

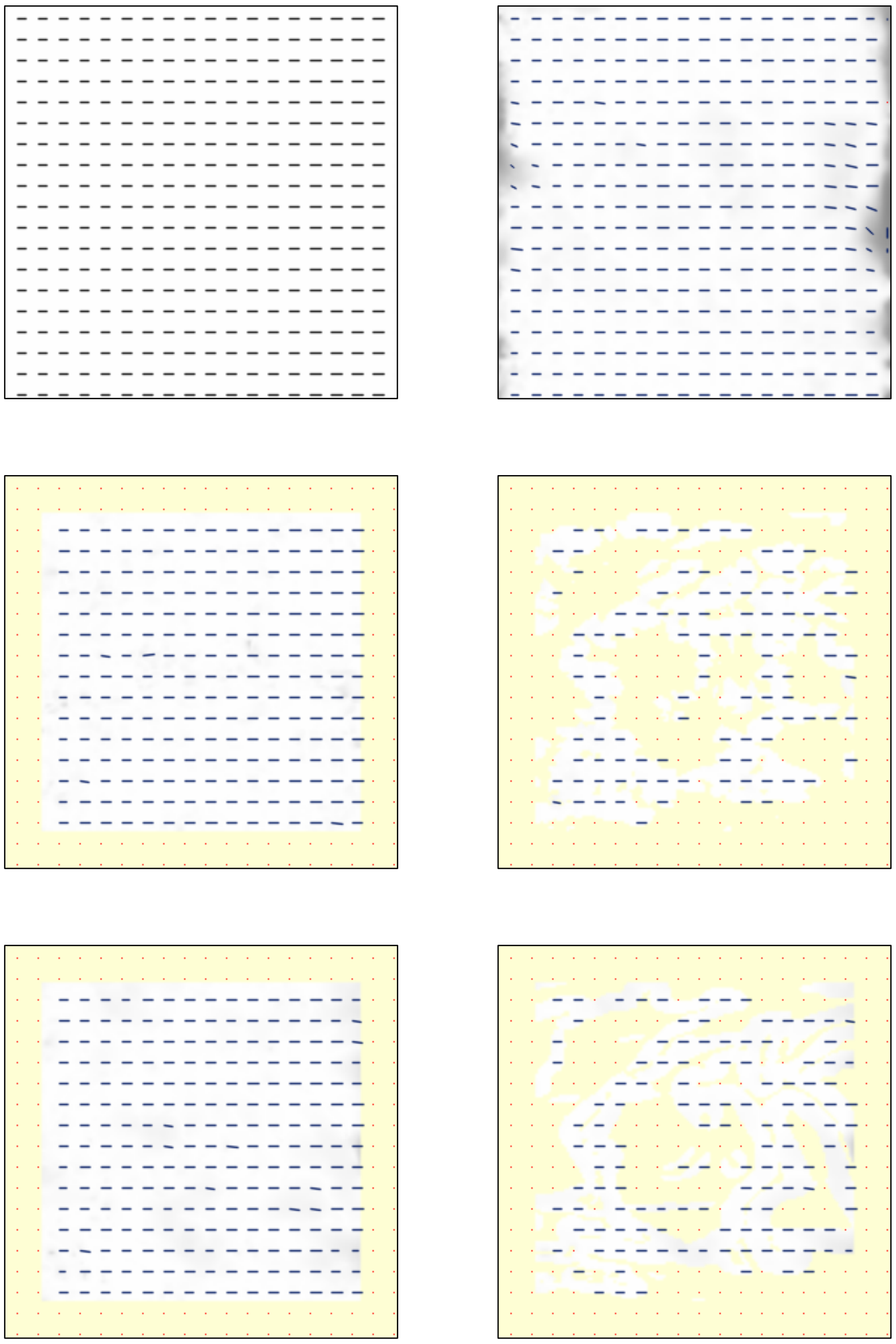


**Figure A.13** Results of the Translating Tree Sequence: The correct flow (top left), the CDWT-ME (top right), Lucas-Kanade with no thresholding (middle left), Lucas-Kanade with threshold = 1.0 (middle right), Horn-Schunck with no threshold (bottom left), and Horn-Schunck with threshold = 5.0 (bottom right).

### A.2.6 The Diverging Tree Sequence

The diverging tree sequence is a more difficult sequence than the translating tree sequence. This is due to the complexity of diverging motion over the translating motion. Hence, the overall errors have increased for all methods. The LK and HS showed similar performance. The CDWT-ME remained behind their performances. Here, again, the uniform diverging motion favored the LK and HS algorithms as they rely on the gradient computations. The flow images are shown in Figure A.14 and the corresponding results are given in Table A.6.

**Table A.6** Results of the Diverging Tree Sequence.

| ***Algorithm*** | ***Density*** | | ***Angle Error*** | | ***Magnitude Error*** | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 19934 | 88.6 | 5.6783 | 7.7700 | 0.0972 | 0.1374 |
| CDWT-ME (Conf. Thresh. = 0.95) | 11479 | 51.0 | 4.1904 | 5.0814 | 0.0736 | 0.1048 |
| Lucas-Kanade (Tau = 1.0) | 8090 | 36.0 | 1.9911 | 1.9955 | 0.0286 | 0.0315 |
| Lucas-Kanade (Tau = 0.0) | 14884 | 66.2 | 2.5295 | 2.5007 | 0.0360 | 0.0422 |
| Horn-Schunck (Tau = 5.0) | 7967 | 35.4 | 2.1964 | 2.2926 | 0.0342 | 0.0500 |
| Horn-Schunck (Tau = 0.0) | 14884 | 66.2 | 2.2867 | 2.3051 | 0.0358 | 0.0483 |
| CDWT-ME (at LK Tau = 1.0) | 8044 | 35.8 | 3.4912 | 2.6272 | 0.0551 | 0.0533 |

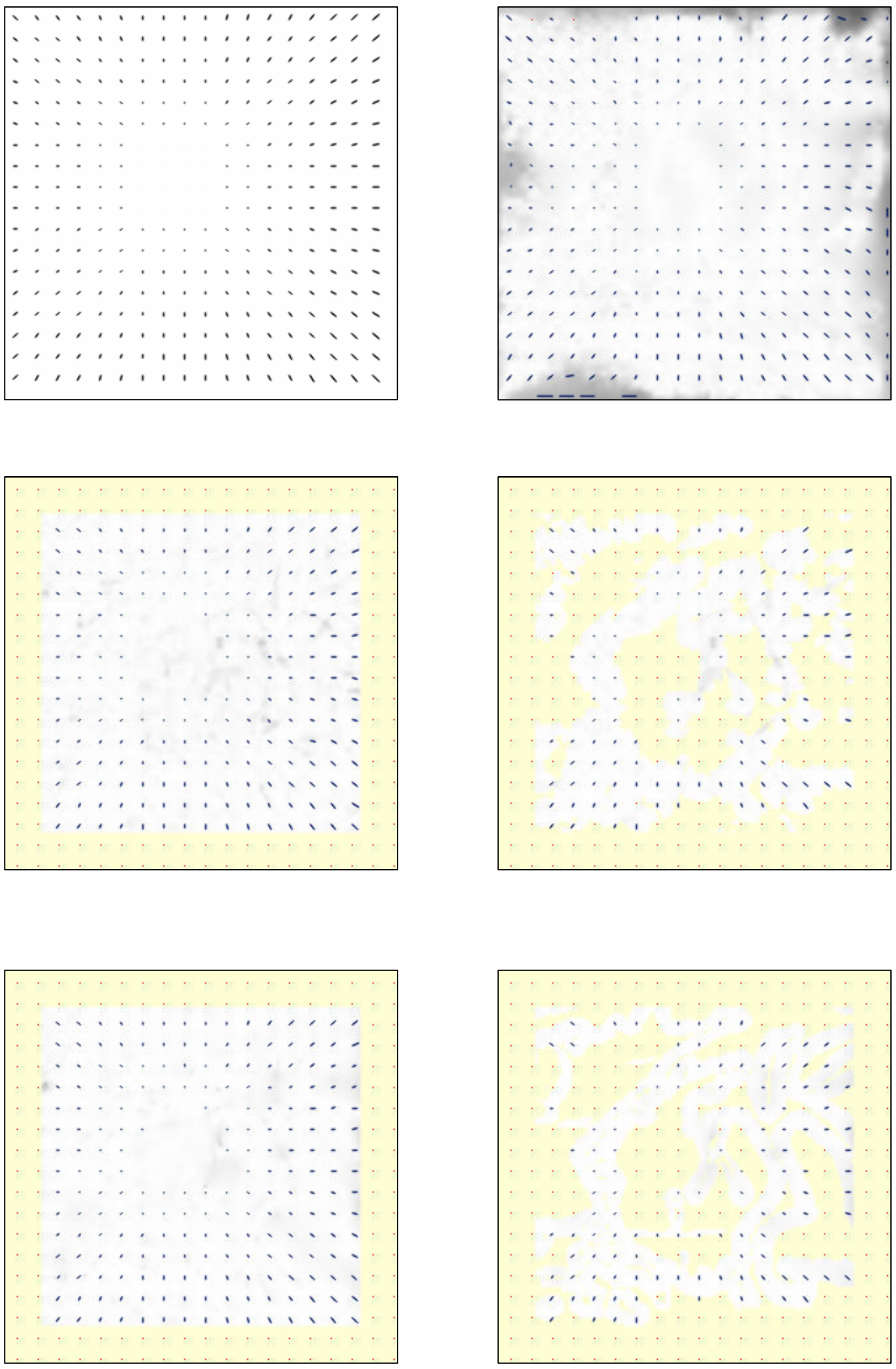


**Figure A.14** Results of the Diverging Tree Sequence: The correct flow (top left), the CDWT-ME (top right), Lucas-Kanade with no thresholding (middle left), Lucas-Kanade with threshold = 1.0 (middle right), Horn-Schunck with no threshold (bottom left), and Horn-Schunck with threshold = 5.0 (bottom right).

### A.2.7 The Yosemite Sequence

This is the most difficult sequence. It contains both translational and divergent motion. The motion of the clouds is translational while the rest of the motion is divergent. Also there are intensity changes in the clouds and in addition the clouds have very low texture. The error results are given in Table A.7 and the corresponding flow images are shown in Figure A.15.

First, examining the thresholded values we see that the LK and HS were more accurate than the CDWT-ME. However, if we look at the overall results (no thresholding) we see that for the whole image, the CDWT-ME has performed better than the other methods. Second, examining the density values, we see that the CDWT-ME has created a more dense motion field than the other ones.

Taking a look at the flow images it becomes more evident that after thresholding, the LK and HS have produced confident flows for only a small part of the image, whereas the CDWT-ME has covered a greater part. For the top part, where the translating clouds are, the flow field creates by HS and LK is irrelevant. However, the CDWT-ME has produced a quite reasonable flow field.

**Table A.7** Results of the Yosemite Sequence.

| ***Algorithm*** | ***Density*** | | ***Angle Error*** | | ***Magnitude Error*** | |
|---|---|---|---|---|---|---|
| | *Total* | *Percent* | *mean* | *std* | *mean* | *std* |
| CDWT-ME (Conf. Thresh. = 0.00) | 73888 | 92.8 | 7.7314 | 12.5191 | 0.2950 | 0.4192 |
| CDWT-ME (Conf. Thresh. = 0.95) | 39104 | 49.1 | 6.8730 | 12.6492 | 0.2390 | 0.3553 |
| Lucas-Kanade (Tau = 1.0) | 25669 | 32.2 | 4.4875 | 12.1617 | 0.0952 | 0.2347 |
| Lucas-Kanade (Tau = 0.0) | 64512 | 81.0 | 9.9308 | 17.8978 | 0.4968 | 1.0809 |
| Horn-Schunck (Tau = 5.0) | 21216 | 26.6 | 5.4825 | 11.3074 | 0.1407 | 0.2712 |
| Horn-Schunck (Tau = 0.0) | 64512 | 81.0 | 9.5685 | 15.8002 | 0.4548 | 0.9201 |
| CDWT-ME (at LK Tau = 1.0) | 25554 | 32.1 | 6.2745 | 13.0863 | 0.1378 | 0.3044 |

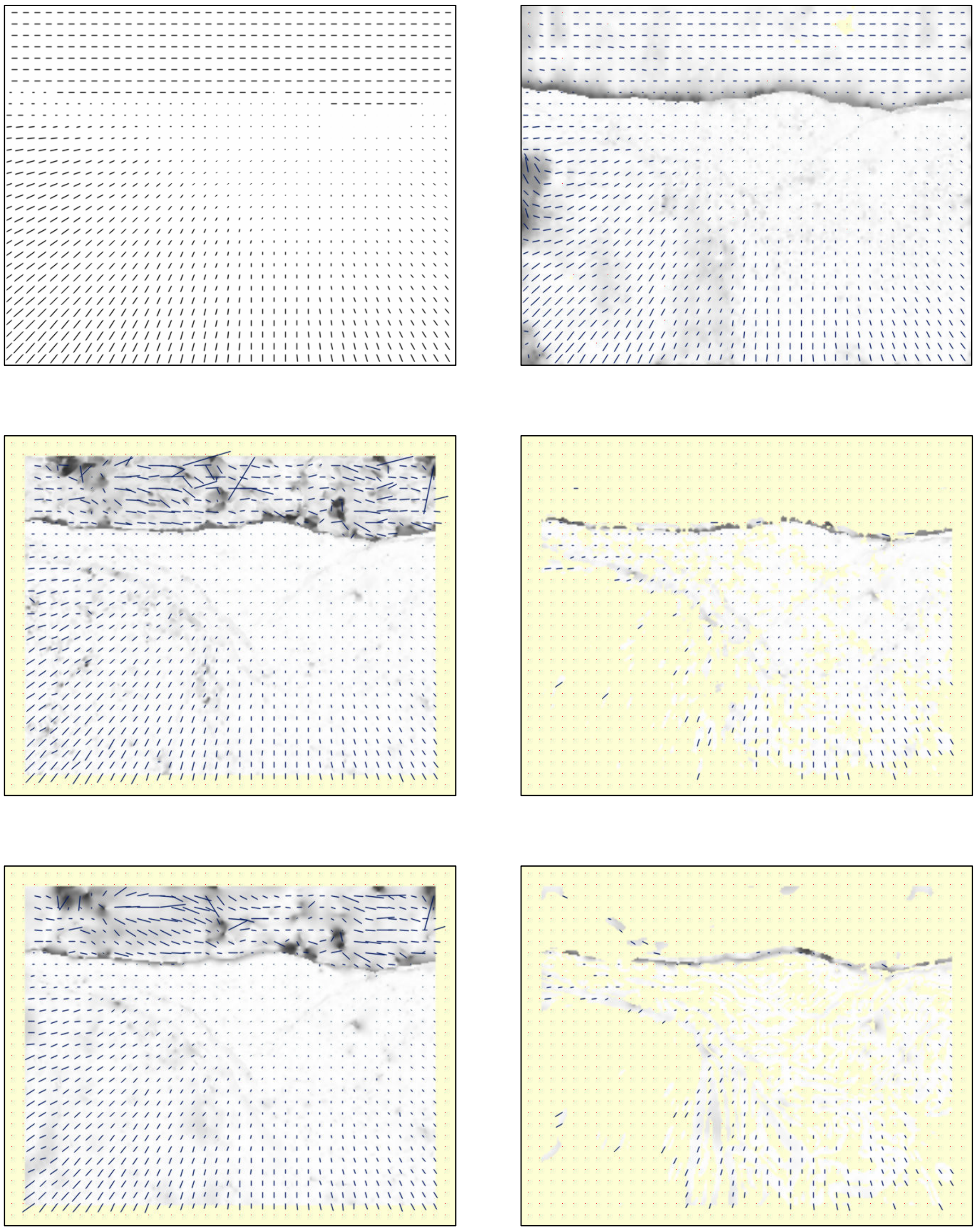


**Figure A.15** Results of the Yosemite Sequence: The correct flow (top left), the CDWT-ME (top right), Lucas-Kanade with no thresholding (middle left), Lucas-Kanade with threshold = 1.0 (middle right), Horn-Schunck with no threshold (bottom left), and Horn-Schunck with threshold = 5.0 (bottom right).

## A.2.8 The Rotating Rubik Cube Sequence

The rotating cube is a real sequence. The resulting flow fields generated by CDWT-ME and LK algorithms are shown in Figure A.16. Examining the flow fields we see that both the CDWT and the LK seem to have produced similar results. However, due to the uniform background, there is also flow assigned to the background where there is no actual flow.

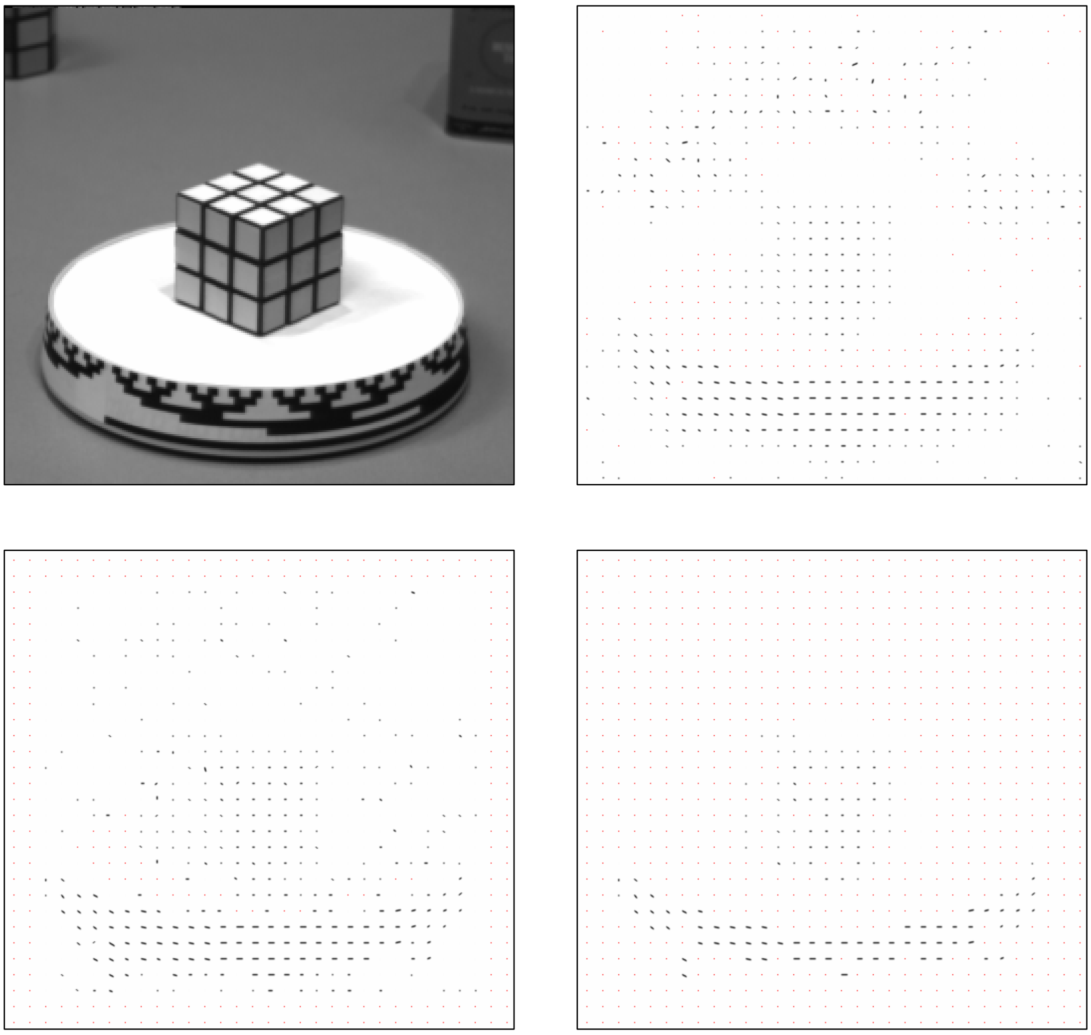


**Figure A.16** Results of the Rotating Rubik Sequence: A frame from the original sequence (upper left), the CDWT-ME (upper right), Lucas-Kanade with no thresholding (lower left), Lucas-Kanade with threshold = 1.0 (lower right).

### A.2.9 The Hamburg Taxi Sequence

In the Hamburg Taxi Sequence, there are four different moving objects. Regarding the pedestrian walking on the upper left corner, both algorithms have missed this. For the black car on the left, CDWT-ME has produced a slightly better flow field. Regarding the white taxi turning, both algorithms produced similar results. Finally, for the vehicle on the right, which is partly covered by the tree, the CDWT-ME have produced a more reasonable flow field than the LK algorithm. Looking at the thresholded flow for the LK, only the flow of white taxi remains confident. The resulting flow fields are shown in Figure A.17.

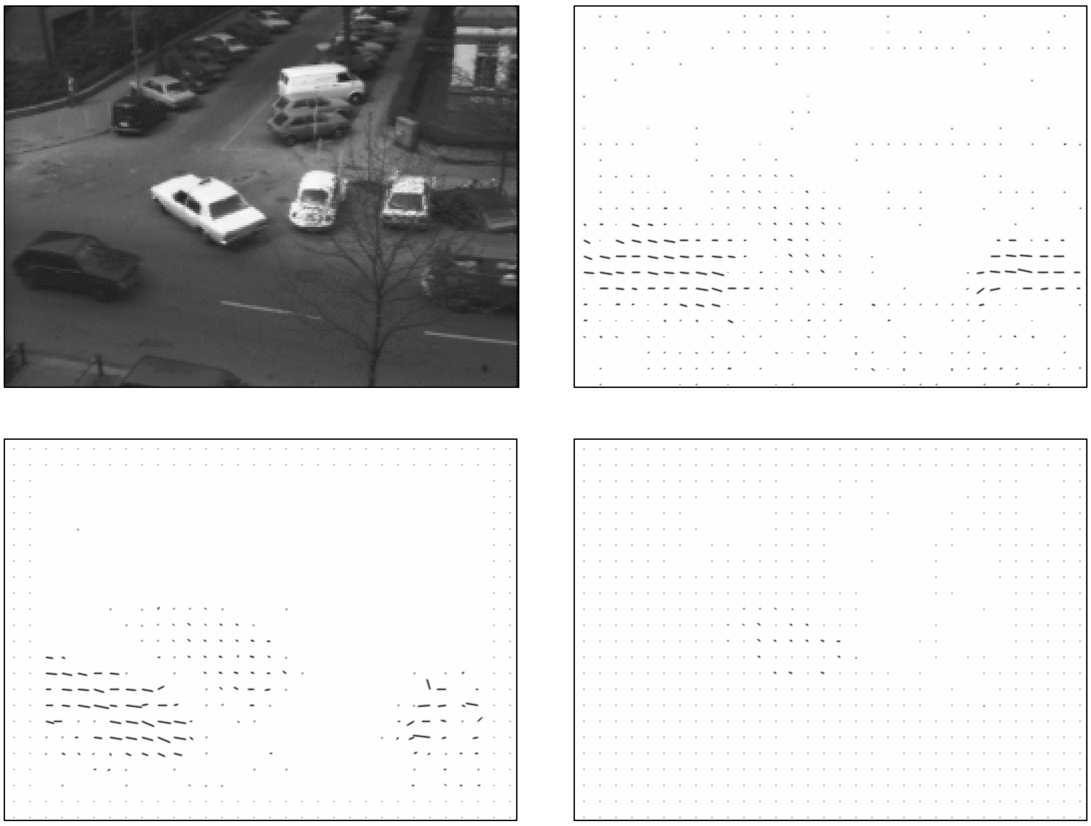


**Figure A.17** Results of the Hamburg Taxi Sequence: A frame from the original sequence (upper left), the CDWT-ME (upper right), Lucas-Kanade with no thresholding (lower left), Lucas-Kanade with threshold = 1.0 (lower right).

### A.2.10The NASA Sequence

The NASA Sequence is a divergent sequence. The flow field produced by the CDWT-ME and the LK algorithms is given in Figure A.18. The flow field generated by the CDWT-ME is very smooth and more reasonable than the flow generated by the LK algorithm, which has some irrelevant flow regions.

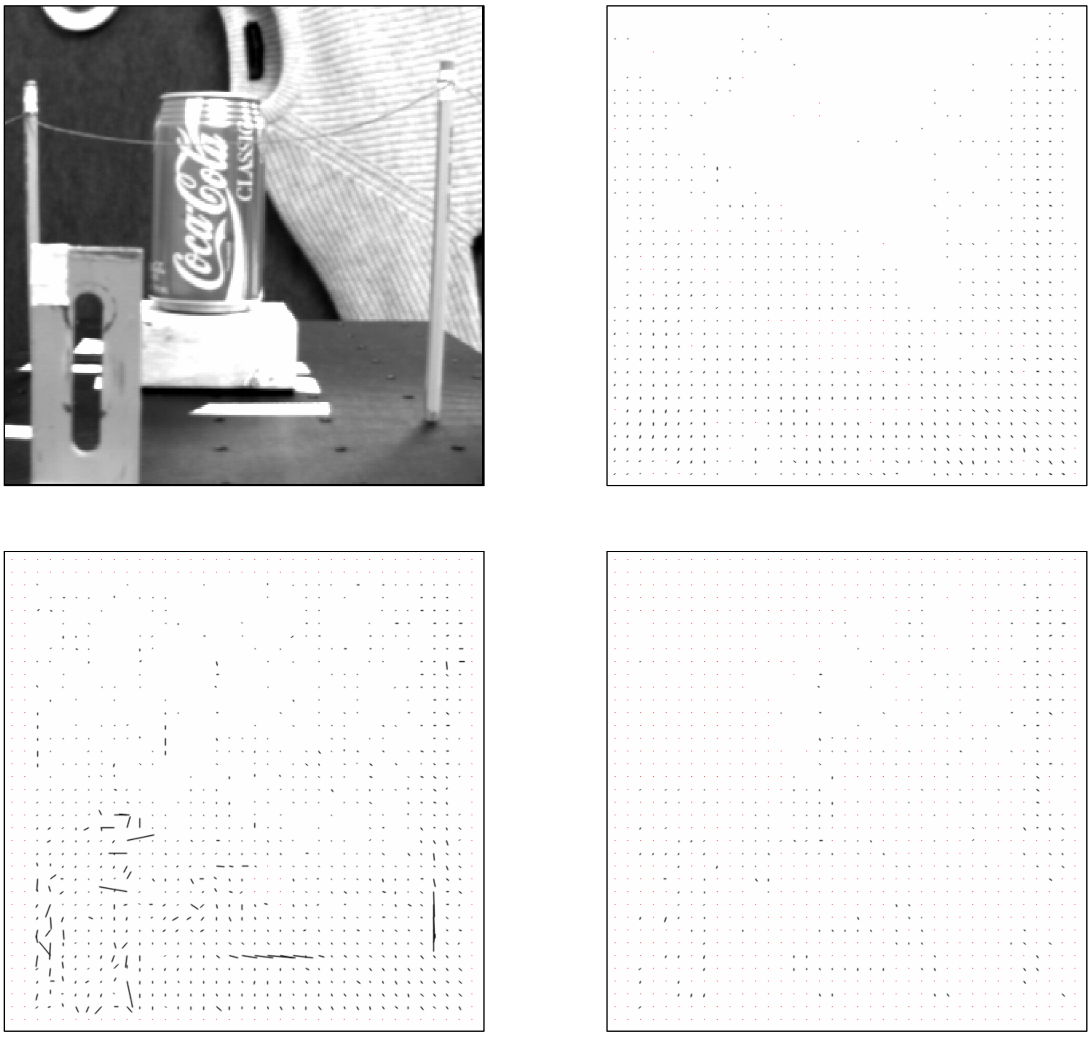


**Figure A.18** Results of the NASA Sequence: A frame from the original sequence (upper left), the CDWT-ME (upper right), Lucas-Kanade with no thresholding (lower left), Lucas-Kanade with threshold = 1.0 (lower right).

### A.2.11The SRI Sequence

The SRI Sequence is a difficult sequence since the camera is moving in one direction producing large relative motions in the front and small motions in the back of the image. There are different levels of motion within each other. The flow field generated by the CDWT-ME and the LK algorithms are given in Figure A.19.

Comparing the flow fields, the CDWT-ME has produced a smoother and highly reasonable flow field compared to the LK algorithm. There are quite a lot irrelevant flows some of which remain even after thresholding.

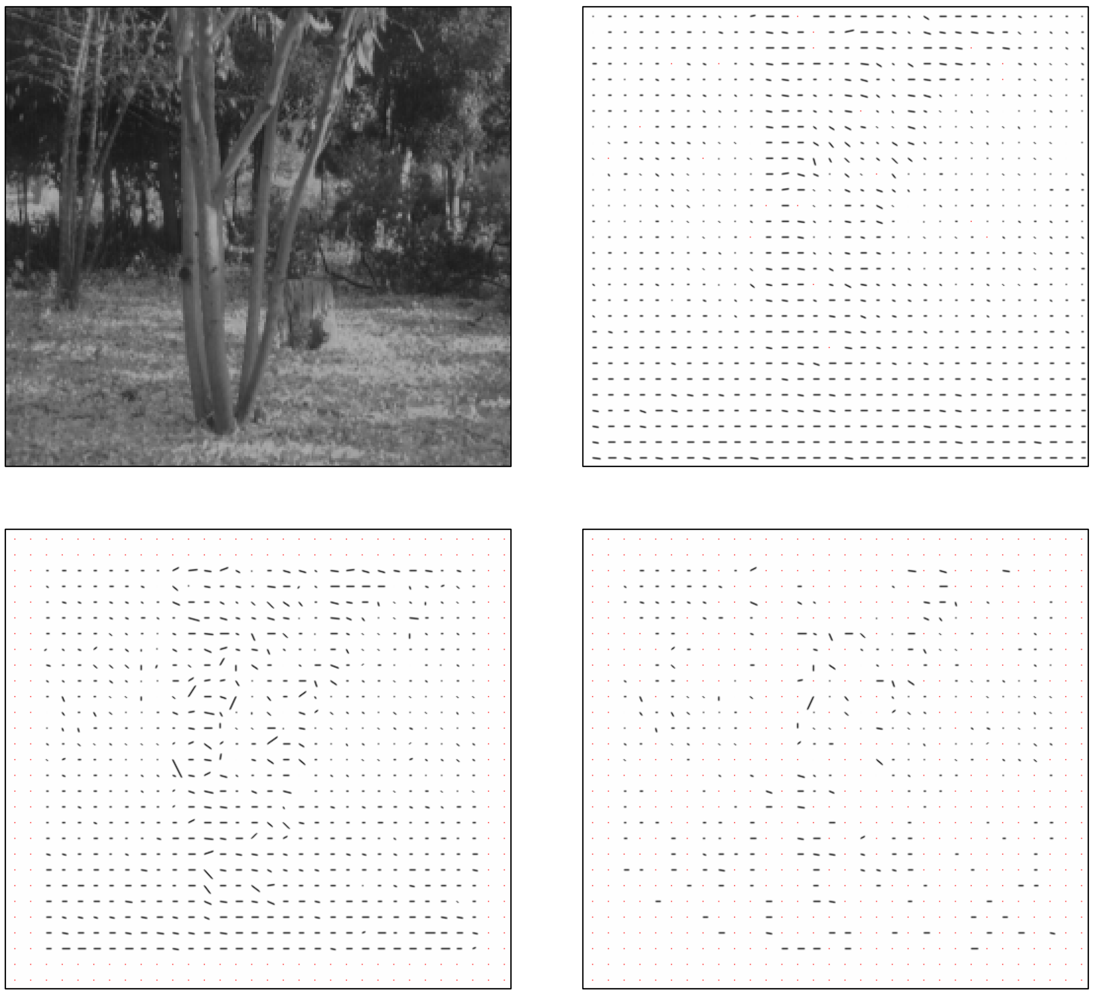


**Figure A.19** Results of the SRI Sequence: A frame from the original sequence (upper left), the CDWT-ME (upper right), Lucas-Kanade with no thresholding (lower left), Lucas-Kanade with threshold = 1.0 (lower right).

# VITA

Şener Yılmaz was born in Berlin, Germany on September 02, 1973. He received his B.Sc. degree in Electrical and Electronics Engineering from the Bilkent University, Ankara, Turkey, in 1996. He received his M.Sc. degree in Electrical and Electronics Engineering from the Middle East Technical University, Ankara, Turkey, in 1999 with a thesis entitled "An Automatic Target Detector and Classifier for a Security and Surveillance System."

Besides his graduate studies he is working as a Lead Software Design Engineer in the Image Processing Department of ASELSAN Inc., Ankara, Turkey, since 1996. He is also leading the Image Processing Algorithms Design Group in ASELSAN. He has designed and developed the control and video processing software, including automatic target detection algorithms, for a day and night infrared surveillance system. He has designed and developed the video processing software for a second generation thermal imaging system. He has designed and developed the video processing and tracking software for an airborne electro-optical targeting system. Recently, he is leading the software team of a mission and safety-critical software project for airborne platforms.

His areas of interest include infrared image and video processing (front-end and back-end), image and video enhancement, motion estimation, video compression, image understanding, machine vision, automatic target detection, and automatic target tracking as well as embedded software development, real-time operating systems, software safety, mission- and safety-critical software development, software reliability, software quality, and software lifecyle processes.