Information Fusion for Improved Motion Estimation

Andrew Mark Peacock

A thesis submitted for the degree of Doctor of Philosophy.
The University of Edinburgh.
May 2001

Abstract

Motion Estimation is an important research field with many commercial applications including surveillance, navigation, robotics, and image compression. As a result, the field has received a great deal of attention and there exist a wide variety of Motion Estimation techniques which are often specialised for particular problems. The relative performance of these techniques, in terms of both accuracy and of computational requirements, is often found to be data dependent, and no single technique is known to outperform all others for all applications under all conditions. Information Fusion strategies seek to combine the results of different classifiers or sensors to give results of a better quality for a given problem than can be achieved by any single technique alone. Information Fusion has been shown to be of benefit to a number of applications including remote sensing, personal identity recognition, target detection, forecasting, and medical diagnosis.

This thesis proposes and demonstrates that Information Fusion strategies may also be applied to combine the results of different Motion Estimation techniques in order to give more robust, more accurate and more timely motion estimates than are provided by any of the individual techniques alone.

Information Fusion strategies for combining motion estimates are investigated and developed. Their usefulness is first demonstrated by combining scalar motion estimates of the frequency of rotation of spinning biological cells. Then the strategies are used to combine the results from three popular 2D Motion Estimation techniques, chosen to be representative of the main approaches in the field. Results are presented, from both real and synthetic test image sequences, which illustrate the potential benefits of Information Fusion to Motion Estimation applications.

There is often a trade-off between accuracy of Motion Estimation techniques and their computational requirements. An architecture for Information Fusion that allows faster, less accurate techniques to be effectively combined with slower, more accurate techniques is described.

This thesis describes a number of novel techniques for both Information Fusion and Motion Estimation which have potential scope beyond that examined here. The investigations presented in this thesis have also been reported in a number of workshop, conference and journal papers, which are listed at the end of the document.

Declaration of originality

I hereby declare that the research recorded in this thesis and the thesis itself was composed and originated entirely by myself in the Department of Electronics and Electrical Engineering at The University of Edinburgh.

Andrew Mark Peacock

Acknowledgements

Firstly, I would like to thank my two supervisors Dr John Hannah and Dr David Renshaw for their invaluable advice, guidance and support throughout the last 3 years.

I would also like to thank Dr Pete Edwards for discussions on Estimation Fusion and for introducing me to the Paper Curl data set, which is proving a useful alternative problem for the Estimation Fusion ideas presented in this thesis.

Thanks also go to Dr Andy Goater, Dr Richard Lee and Prof. Ron Pethig for bringing the electrorotation application to my attention and supplying the sample video sequences.

I would like to thank Shinichiro Matsunaga for his comments on the novel block matching update strategies and for providing the car following test sequences. Thanks also to Chris Haworth for discussions on this work. Chris has since extended this work by looking in more detail at the robustness of the different update strategies.

My thesis has been kindly supported by EPSRC under studentship award number 98318229.

Contents

List of figures

List of tables

Acronyms and abbreviations

1DFS	1 Dimensional Full Search
2DFS	2 Dimensional Full Search
AD	Absolute Difference
AE	Accuracy-Efficiency
ABMA	Adaptive Block Matching Algorithm
BMA	Block Matching Algorithm
CAMUS	An example Motion Estimation strategy [1]
CC	Correlation Coefficient
DCT	Discrete Cosine Transform
EDS	Element Difference Signal
FDS	Frame Difference Signal
FSA	Full Search Algorithm
HORN	An example Motion Estimation strategy [2]
IR	Infrared
LUT	Look Up Table
MAD	Mean Absolute Difference
MPE	Minimum Probability of Error
MPI	Message Passing Interface
MRD	Multi Resolution Decomposition
NCC	Normalised Correlation Coefficient
pel	Picture Element
SD	Squared Difference
SE	Squared Error
TAD	Thresholded Absolute Difference
TSE	Thresholded Squared Error
URAS	An example Motion Estimation strategy [3]

Nomenclature

Δ	Threshold value used
λ	Weighting term from HORN
$\lambda()$	Cost function
$\mu()$	Fuzzy membership function
Φ	State transition matrix
θ	Direction of motion
A	Class/Set of elements
B	Class/Set of elements
$Bel()$	Belief function
D	Similarity/distortion
E_M	Error due to motion gradient constraint
E_S	Error due to smoothness constraint
$f()$	Comparison function
H	Set of hypotheses
H	Matrix converting from state to observation space
h	Hypothesis
$I()$	Pel intensity function
J	Euclidean distance
K	Kalman gain
L	Number of potential search locations
$m()$	Mass function (basic probability number)
$\mathbf{m}$	Best match block vector
$\mathbf{m}$	Mean motion vector
M	Number of hypotheses
N	Number of individual classifiers/techniques
N	Block width
$p()$	Probability distribution
P	Probability $P()$
$P()$	Probability

$Pls()$	Plausibility function
$\mathbf{q}$	Measurement Error
Q	Measurement error covariance matrix
r	Magnitude of motion
$\mathbf{r}$	State vector
R	Number of vectors
R	Risk
R	Dynamic process noise covariance matrix
S	Estimate error covariance matrix
t	Time
t	Time range
^{T}	Denotes transpose
u	Horizontal motion component
v	Vertical motion component
w	Spatial search window size
$\mathbf{x}$	Vector of individual observations/decisions
x	Individual observation/decision
x	Pel horizontal location index
$\mathbf{X}$	Observation space
X	Region in observation space
y	Value from continuous range
y	Pel vertical location index
$\hat{y}()$	Estimate of y
Y	Continuous range

Chapter 1
Introduction

1.1 Introduction

Information Fusion techniques seek to combine information from different sources so as to provide information of better quality than can be provided by any of the individual sources alone [4]. Information Fusion [5–11] has been shown to improve the quality of results for a wide variety of classification, detection and estimation problems including remote sensing [12–15], personal identity recognition [16–19], handwriting recognition [20], target detection [21–24], image segmentation [25], forecasting [26] and medical imaging [27, 28]. Motion Estimation from image sequences is an important research field with many applications including image sequence compression [29–32], surveillance [33, 34], navigation [1, 35] and tracking [36, 37].

This thesis investigates the application of Information Fusion to combine the results of Motion Estimation techniques and shows that this can lead to more accurate, robust and timely motion estimates. The remainder of this chapter is structured as follows: Section 1.2 presents motivations for this work and Section 1.3 outlines the main contributions of the thesis. Section 1.4 describes the structure of the thesis and the chapter contents. A summary of the chapter is given in Section 1.5.

1.2 Motivations

In common with most Image Processing algorithms, Motion Estimation techniques rely on assumptions about the image sequences and when these assumptions are broken the accuracy of the motion estimates can be poor. A typical assumption, for example, is that the appearance of an imaged object will not change over time and motion. Different techniques can have different underlying assumptions and use these assumptions in different ways, and so their relative performance may vary depending on the image sequence.

Figure 1.1(a) shows an example frame from a real test sequence in which a planar object (a photograph) moves horizontally at constant velocity from right to left across the field of view.

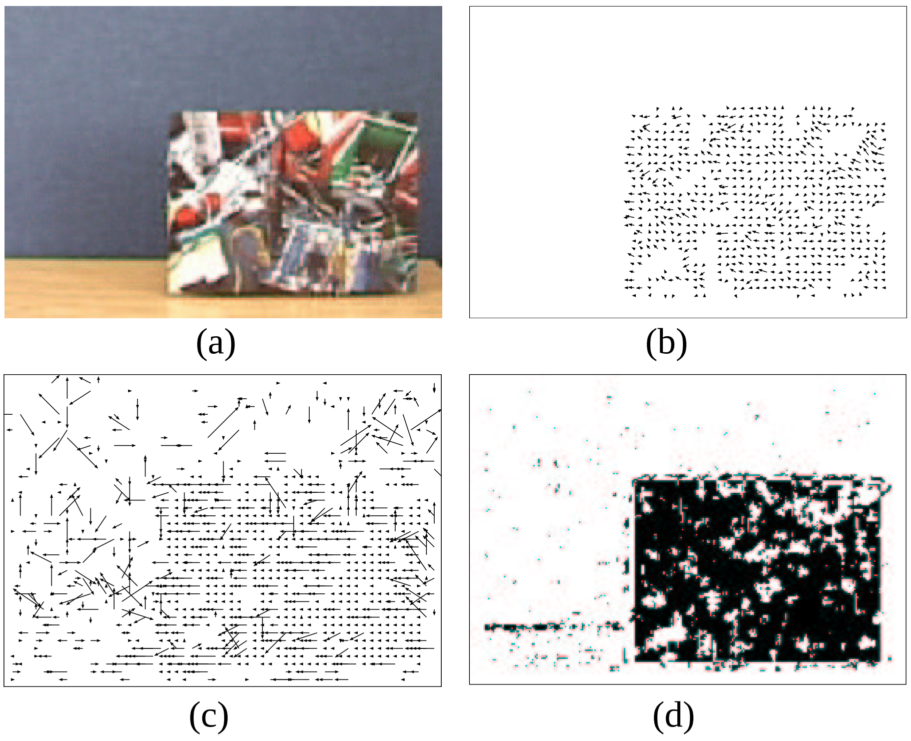


Figure 1.1: *Motion estimates from two different techniques applied to a test sequence in which a planar object moves at constant velocity from right to left in front of a motionless background. (a) example frame from the test sequence, (b) results from technique A, (c) results from technique B. d) A comparison between the performances of the techniques. White pels indicate that technique A gave smaller errors, black indicates that technique B gave smaller errors.*

Figures 1.1(b) and 1.1(c) illustrate results from applying two different Motion Estimation techniques, A and B, to this test sequence. These Motion Estimation techniques will be fully described in Chapter 3. Qualitatively, it can be seen that the relative performance of these techniques is data dependent, with A performing better on the background and B giving more accurate results on the foreground object.

As this test sequence is very simple, it is easy to manually determine the true motion and hence obtain a quantitative measure of performance. Performance is judged using an error measure which will be described in Chapter 3 against the ground truth motion field for the test sequence. Figure 1.1(d) illustrates the relative performance of the techniques by showing pels where B performed better in black and pels where A performs better in white. Quantitively, A gives more accurate motion vectors for only 22% of pels on the foreground photograph but for 97% of the background pels.

There exist a variety of Motion Estimation strategies including correlation methods [1, 34, 37–40], differential methods [2, 3, 41] and frequency domain approaches [42–46]. These different strategies make different assumptions and so the relative performance of these techniques in terms of both accuracy and computational requirements is often data dependent. As Motion Estimation is an important component of many applications, there is a need to find techniques that can cope well on a range of problems. This thesis is motivated by the desire to combine different Motion Estimation strategies so as to give good results over a more general range of problems than can be coped with by any single strategy alone.

1.3 Contributions

The main contribution of this thesis is to identify, expand and apply appropriate Information Fusion strategies for representing and combining motion estimates [47–49]. This research has more general applications outside Motion Estimation and has since been applied to combining the results of different Neural Networks [50].

An important problem when fusing motion estimates is that different techniques often have different computational requirements. There is often a trade-off between the speed and accuracy of Motion Estimation strategies. This thesis also proposes and demonstrates a fusion architecture which can help solve this problem [51, 52].

The background chapters of this thesis give an overview of Information Fusion and Motion Estimation research. These chapters also present novel techniques for use in Motion Estimation in the frequency domain [53] and for tracking objects[54].

1.4 Structure

The structure of this thesis is as follows:

- Chapter 2 gives an introduction to Information Fusion research. Applications and strategies for Information Fusion are described, and an appropriate strategy for fusing motion estimates is identified.
- Chapter 3 gives a brief overview of Motion Estimation. Motion Estimation algorithms used in the remainder of the thesis to illustrate the fusion strategies are described and examples of their application to synthetic and real data sequences are presented.
- Chapter 4 presents an illustrative example where Information Fusion techniques are applied to combine estimates of the frequency of rotation of spinning cells. The motion estimates in this example are scalar values rather than the 2D motion vectors considered by the remainder of the thesis.
- Chapter 5 considers the hypothesis that Information Fusion can help provide more accurate and robust 2D motion estimates. Novel techniques for improving the robustness of Information Fusion are proposed and applied. Empirical results from synthetic and real data are presented to support the hypothesis.
- Chapter 6 considers the hypothesis that Information Fusion can help provide more timely motion estimates. To this end it is shown that Information Fusion can help to overcome the accuracy-efficiency trade-off problem in Motion Estimation.
- Chapter 7 presents conclusions and a summary of the thesis, a discussion of its limitations and suggestions of topics for future research.

1.5 Summary

The theme of this thesis is the application of Information Fusion to improving the quality of Motion Estimation results. This is motivated by the need for more accurate and robust motion estimates, by existing research which has shown that Information Fusion can significantly improve the quality of results in other fields, and by the observation that the relative performance of different Motion Estimation algorithms is often data dependent.

This thesis makes a number of contributions to both the Information Fusion and Motion Estimation research fields. Empirical results are presented which confirm that Information Fusion can combine the results of different Motion Estimation algorithms to give more accurate, robust and timely motion estimates.

Chapter 2
Information Fusion

2.1 Introduction

Results from different sensors or classifiers can often be combined to give estimates of a better quality than could be obtained from any of the individual sources alone. Luo and Kay give a comprehensive survey of the Information Fusion field in [5], their paper also appears with a collection of other fusion survey papers in [7]. An introduction to the field with particular interest in the influence of defence research is given by Hall and Llinas in [9]. Surveys of multisensor decision fusion strategies are given by Dasarathy in [6] and by Varshney in [10]. An overview of Millimeter-Wave and Infrared Multisensor Design is given by Klein in [24]. Bloch [8] reviews a variety of combination operators and classifies them depending on their behaviour.

This chapter presents an overview of the Information Fusion field. The structure of this chapter is as follows: Section 2.2 gives example applications of Information Fusion and Section 2.3 describes some popular fusion strategies identified from these examples. The appropriateness of these strategies for Motion Estimation is discussed in Section 2.4, which identifies an appropriate strategy for combining motion estimates. This strategy is described more fully in Section 2.5 and a summary of the chapter is given in Section 2.6.

2.2 Applications

Information Fusion has a wide variety of applications including remote sensing [12–15], personal identity recognition [16–19], handwriting recognition [20], target detection [21–24], image segmentation [25], forecasting [26] and medical imaging [27, 28]. As well as being applied to combine information for different purposes, Information Fusion can be applied at the sensor (data) level, pel level, feature level and decision level, with the fused decision either at the same level or at a higher level [10, 55]. To a certain extent the application dictates the choice of fusion strategy. Wald [4], who reports the results of a European working group set up to harmonise reference terms in the field, gives the following definition of data fusion:

> Data fusion is a formal framework in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; the exact definition of "greater quality" will depend on the application.

The application considered by this thesis is that of Motion Estimation. Although there has been some interest in combining Stereo Vision techniques with Motion Estimation [56, 57], there do not appear to be examples of Information Fusion techniques being applied to Motion Estimation in the literature, and so the remainder of this section briefly discusses other applications in order to identify the methods they use. The applications considered are: personal identity recognition, remote sensing, image fusion and training neural networks.

2.2.1 Personal Identity Recognition Systems

The problem of establishing the identity of individuals from their biometric features has received a great deal of attention [16, 18, 19, 58]. Such systems overcome many of the disadvantages of password based secure access systems. Typical biometric features which can be used for identification include voice, iris, fingerprints, gait and face appearance. The additional information that is available to multi-modality approaches can lead to more robust identity recognition systems. Furthermore, the use of multiple modalities can reduce the risk of malevolent individuals gaining access to a protected system by imitating authorised users, for example by playing a voice recording to fool a voice recognition system. As well as combining the evidence from multiple biometric modalities in a feature level strategy, the classifications given by different techniques can themselves be combined in a decision level strategy.

The identity recognition problem is an M hypothesis classification problem where the M hypotheses are the individuals in a client database. A simpler version of the problem is that of personal identity authentication, where the identity of an individual is proposed and must be verified. This is a binary hypothesis problem with the hypotheses of either accepting or rejecting the proposed identity.

Chatzis *et al.* [16] use clustering algorithms and fuzzy set theory to combine the decisions of single modality personal identity authentication algorithms. The fused decisions are reported to have significantly fewer errors than the individual decisions. Ben-Yacoub *et al.* [19] combine the classifications of a voice recognition system and a face recognition system in their identity authentication approach. They report that a Bayesian based fusion strategy using the Minimum Probability of Error cost function and an Artificial Neural Network strategy yield the best results. Kittler *et al.* [17] use a Bayesian [59] based fusion methodology to combine evidence from comparing the image of an individual's face against a number of different images of the candidate held on a database. They report a 40% reduction in error rate over using just a single image per candidate database. This work is extended to a full identity recognition system in [18].

Speaker recognition is itself often treated as an information fusion problem, where by the speech signal is divided into small subbands and individual decisions made for each subband are combined [60]. Higgins *et al.* [58] describe a person identification system based on this idea which uses a Bayesian based fusion strategy.

2.2.2 Remote Sensing

Remote sensing from satellites and aircraft is applied to a number of applications, in particular to the classification of land use. Multisensor and hyperspectral images can contain more information than single sensor images. This information can be used to significantly improve the performance of such classifier systems [12–15, 25].

Kermad and Chehdi [25] describe an approach to the location and classification of seaweed using an airborne multispectral imager. They use a decision level fusion strategy where the individual bands are first independently segmented by multi-thresholding. Bands found to have similar results are grouped and class representatives chosen. These segmentations are then combined using clustering algorithms. Hegárat-Mascle *et al.* [15] also fuse land classification results from different sensors. They use Dempster-Shafer evidence theory [61] to combine classifications from individual sensors and report significant improvements over single sensor results. Lee *et al.* [12] compare the performances of probabilistic and Dempster-Shafer fusion strategies applied to combining different sources of remote-sensing data. They conclude that both strategies lead to improved classification accuracy and that there are many similarities between them.

2.2.3 Image Fusion

In Image Fusion, multiple images are combined into a single image. The individual images may be formed by different sensors, from different viewpoints, or by a single sensor at different times. Image Fusion is an increasingly important research area with many applications including medical imaging [28], surveillance [62] and defence [63]. A survey of image fusion strategies is given by Zhang and Blum in [64]. The improved "quality" in Wald's [4] sense relates to the clarity and amount of information in the fused image; Xydeas and Petrovic [65] propose an image fusion performance metric based on the preservation of edge information from the individual images.

Differences between the sensor positions and in imaging characteristics require the individual images to be registered before they can be fused [66]. Physical registration strategies use selected lenses and mirrors to separate the different signals. The design of dual aperture sensors for millimetre radar and infrared is described by Klein in [24]. Toet *et al.*[63] used a germanium mirror to separate wavelengths in their dual aperture Forwards Looking Infrared / visible light sensor. Algorithmic registration strategies identify points of correspondence between the images and use these to determine the parameters of an appropriate transform.

In controlled environments, such as medical imaging, it may possible to use artificial reference points which are visible to all sensors. Matsopoulos *et al.* [28], for example, registered Magnetic Resonance and X-ray Computed Tomography images by placing materials evident to both sensors on the subject. Li *et al.* [67, 68] proposed an automatic visual / infrared registration system based on intensity contours in the images. A consistency checking step was used to discard features which did not appear in both images. Jones *et al.* [62] use halogen spotlights as calibration targets to enable the coregistration of images from visual and thermal infrared cameras.

Objects and features of interest can appear at different scales in the image, depending on both the object size and its distance from the imaging device. To ensure that the fused image captures details at all scales, a Multi-Resolution Decomposition (MRD) can be used. A survey of multiscale based fusion strategies is given by Zhang and Blum in [64]. Early work used a MRD based on the Gaussian Pyramid deconstruction [28, 63, 69], later the wavelet based MRD came into use [70]. The Gaussian Pyramid is a MRD based on successively low pass filtering and subsampling the original image. The initial image forms the bottom level of the pyramid

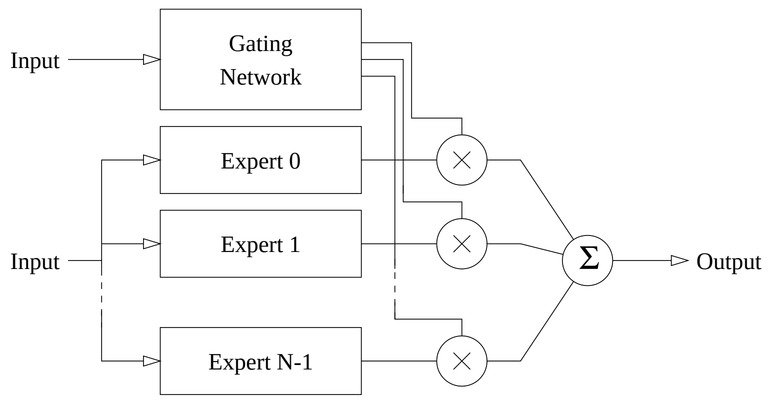


Figure 2.1: *The Mixture of Experts architecture for combining neural networks. A gating network determines the contributions of individual experts.*

with subsequent levels formed by the low pass filtering and subsampling of the previous level. The wavelet transform [71, 72] can also be used to give a multiresolution decomposition of an image [73]. The transform uses successive application of high and low pass filters with subsampling. The wavelet coefficients corresponding to the high pass image capture edge features. By fusing images in the wavelet domain, these perceptually significant features can be preserved.

2.2.4 Training Neural Networks

Fusion strategies are also applied to training Artificial Neural Networks [74]. In the *Committee of Experts* approach, a number of individual networks or *experts* are trained and their outputs combined. This has a number of advantages, particularly because the relative performance of different networks tends to vary over the input space. Edwards *et al.* [75], who use a committee of neural networks to improve quality prediction in paper making, take the average of the individual network outputs as the fused output. Hinton's *Products of Experts* [76] technique for training sets of neural networks uses a combination strategy based on taking the products of individual expert models.

The *Mixture of Experts* [77, 78] approach makes use of the natural decomposition of certain tasks. A set of neural networks are trained on individual subtasks, and the outputs of these experts are combined. A gating network is trained to determine the contributions of the individual experts for a given input pattern. The input to the gating network may or may not be the same as the input to the individual experts. This approach is illustrated in Figure 2.1.

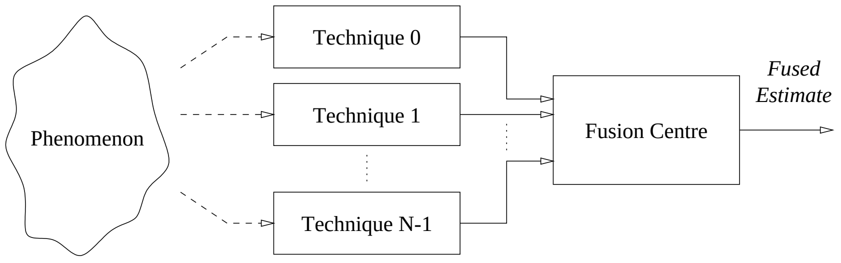


Figure 2.2: *A canonical architecture for Information Fusion.*

A popular alternative to using a gating network is the *competitive learning* approach in which individual experts compete to be selected for particular input patterns. A variety of competitive strategies have been proposed, one of the simplest of these chooses the expert whose weight vector is the smallest Euclidean distance from the input vector [79]. The weights of the winning expert are updated to be more similar to the input vector. McNeill *et al.* [80] compare 4 competitive learning strategies applied to train an autonomous robot to track the motion of a point light source. They conclude that the performance of competitive learning strategies is best when rather than a “winner takes all” strategy the contributions of individual experts to a particular input pattern are weighted.

2.3 Methods of Information Fusion

In the canonical Information Fusion architecture illustrated in Figure 2.2, a phenomenon is observed and processed by a number of individual techniques. Information from the individual techniques is combined at a Fusion Centre. The individual techniques might be sensors [22, 23, 81–83] or different classifiers [18, 84–86]. A wide variety of fusion methods have been proposed and implemented in the literature including committee methods [87, 88], clustering algorithms [16], weighted average, techniques based on Fuzzy set theory [27, 47, 89, 90], Dempster-Shafer evidence theory [12, 15] and Stochastic methods.

A review of decision fusion operators is given by Bloch [8] which classifies operators as context dependent, where global knowledge is taken into account, or context independent where the fusion is dependent only on the individual values. Context independent operators are subdivided into variable behaviour, where the fusion varies depending on the individual values, or constant behaviour.

The remainder of this section gives an overview of the popular approaches to Information Fusion identified in the applications of the previous section: committee methods, clustering methods, methods based on fuzzy set theory and methods based on Dempster-Shafer evidence theory and on Bayesian decision theory.

2.3.1 Committee Methods

A key problem in Information Fusion is how to enable the different information sources to contribute to a result. Vote based decision fusion methods group individual experts or discriminating functions into a set termed a *committee*. In this approach, the individual experts cast votes for the correct hypothesis. A variety of voting rules have been proposed, in the Majority Vote rule the hypothesis with the most votes is chosen. A summary of the background to committee theory is given by Mazurov *et al.* in [87].

Yussoff *et al.* [88] describe a committee based fusion method to combine experts for shot change detection in video sequences. The detection problem is a binary hypothesis problem. They use the Majority Vote rule to combine individual expert decisions for a frame, and describe experiments to find good parameters for the individual experts under this fusion strategy.

Although effective classifiers would be expected to allot high ranks to correct hypotheses, as the number of hypotheses increases, the likelihood of an incorrect choice being ranked top is also increased. By fusing rankings of choices from individual classifiers rather than just the top decisions, the risk of completely rejecting the correct hypothesis can be reduced. Ho *et al.* [86] describe methods to reduce the hypothesis set to as small a subset that still contains the correct hypothesis as possible. Ho *et al.* also discuss class set reordering, where the class rankings are altered to place the true class as close to the top rank as possible.

2.3.2 Clustering Methods

In some classification applications, especially at the feature level, individual results can take values from a range rather than making absolute decisions. For example, the individual personal authentication algorithms used by Chatzis *et al.* [16] give estimates in the range $[0, 1]$ with the limits 0 and 1 indicating rejection and acceptance of the proposed identity respectively. Clustering fusion methods combine such results by forming observation vectors $\mathbf{x}_q$ from the N individual classifiers then by grouping the vectors into M clusters, where M is the number of hypotheses. For an observation $\mathbf{x}_q$, the fused decision is the hypothesis associated with the cluster to which the observation is allocated.

The well known k-means clustering algorithm [91] partitions data into k subsets seeking to minimise the Euclidean distance J between R vectors $\mathbf{x}_q$ and the cluster centroids $\mathbf{y}_i$:

$$J = \sum_{q=0}^{k-1} \sum_{i=0}^{R} ||\mathbf{x}_q - \mathbf{y}_i||^2 \tag{2.1}$$

Chatzis *et al.* [16] use k-means clustering to combine results of personal authentication algorithms by setting $k = M = 2$, with the binary hypotheses of accepting or rejecting the proposed identity.

Clustering is also a popular approach to accumulating evidence for locating and recognising objects [92, 93]. In this approach, features detected in an image are used to strengthen or weaken support for elements in an accumulator table of pose hypotheses. This is also referred to as the Generalised Hough Transform [94].

2.3.3 Fuzzy Set Methods

Often the membership of an element x of a class A is ambiguous. For example, it can be difficult to consistently separate groups of people into classes of “tall”, “medium height” and “small”, as the classifications are not crisply defined. Fuzzy set theory, first introduced by Zadeh [95], associates a “membership function” $\mu_A(x)$ with elements to represent the degree of their membership to a set A. Often $\mu_A(x)$ is constrained to lie in the range $[0, 1]$ so that the closer $\mu_A(x)$ is to 1, the greater the degree of membership. The choice of membership function is usually subjective and parametrised by some confidence measure associated with the classification.

A wide variety of operators for fuzzy sets have been proposed in the literature, and the choice of operator is usually subjective and application dependent [96]. The simple union and intersection operators proposed by Zadeh [95] are illustrative examples whose membership functions, given to classes A and B, are as follows:

$$\mu_{A \bigcup B}(x) = \max(\mu_A(x), \mu_B(x)) \tag{2.2}$$

$$\mu_{A \bigcap B}(x) = \min(\mu_A(x), \mu_B(x)) \tag{2.3}$$

Nejatali and Ciric [27] use fuzzy sets to represent both class membership and sensor noise in their multisensor image fusion approach. They give a simulated example where electrical impedance tomography is applied at different frequencies to detect biological organs in a subject. A membership function based on pel intensity is used to classify pels in the individual images formed by each frequency band. These individual classifications are then combined using an operator which is similar to (2.2) but which also takes account of sensor uncertainty.

Schnatter [97] suggests that the inaccuracy of a measuring process is best expressed by fuzzy rather than stochastic methods. Accordingly, Hong and Wang [89] use fuzzy theory to express uncertainty due to the measuring process in their multisensor fusion strategy based on the Kalman Filter [98].

2.3.4 Dempster-Shafer Evidence Theory

Dempster-Shafer Evidence theory [61] offers an alternative method of dealing with uncertainty in which a *basic probability number* or *mass function* $m(A)$ is associated with every element A of the set of subsets 2^H of the hypothesis space H so that [15]:

$$m(\emptyset) = 0 \tag{2.4}$$

$$\sum_{A \in 2^H} m(A) = 1 \tag{2.5}$$

The mass function $m(A)$ is a measure of the belief exactly in A, the total belief Bel(A) in A is the sum of its mass function and the mass functions of all its subsets B:

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B) \tag{2.6}$$

Another useful function is the Plausibility function Pls(A), which measures the degree to which A is not doubted:

$$\text{Pls}(A) = \sum_{B \bigcap A \neq \emptyset} m(B) \tag{2.7}$$

Plausibility and Belief functions represent the maximum and minimum uncertainty in A. As A may be any subset of H, uncertainty about sets of hypotheses can be represented. Two mass functions m_A and m_B can be combined to give a third mass function m_C using the orthogonal sum, defined as follows [61]:

$$m_C(A) = \frac{\sum_{A_i \bigcap B_j = A} m_A(A_i) m_B(B_j)}{1 - \sum_{A_i \bigcap B_j = \emptyset} m_A(A_i) m_B(B_j)} \tag{2.8}$$

if the denominator is 0 then the orthogonal sum is said not to exist.

Hegárat-Mascle *et al.* [15] use Dempster-Shafer evidence theory in their multi-sensor land classification approach. They describe an unsupervised method for allocating mass functions to hypotheses sets and combine these using the orthogonal sum.

2.3.5 Probabilistic Methods

Bayesian detection theory offers a well understood framework which can be used for Information Fusion in which uncertainty in the individual results is expressed by probabilities. Classical Bayesian detection theory was first extended to the distributed sensor case by Tenney and Sandell [23], who developed optimal decision functions for the individual detectors. However, they did not develop rules for the fusion centre. Chair and Varshney [22] extended the work of Tenney and Sandell by developing an optimal decision fusion strategy for a binary hypothesis detection problem. Thomopoulos *et al.* [81] developed an optimal decision fusion technique using the Neyman-Pearson test. Out *et al.* [99] applied the Bayesian fusion approach to combine predictions from neural networks.

Kittler *et al.* [84, 85, 100] developed a theoretical framework based on Bayesian theory, and

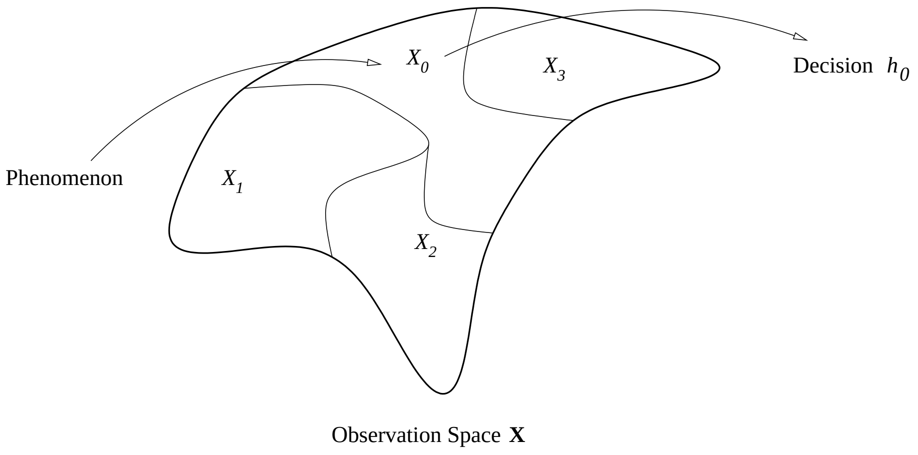


Figure 2.3: *The observation space* $\mathbf{X}$ *is split into disjoint regions* X_i*, so that given an observation* $\mathbf{x}$*, if* $\mathbf{x} \in X_i$ *then hypothesis* h_i *is chosen.*

show that many popular fusion strategies are special cases of this framework. Anandaligam and Chen [26] described a general Bayesian model for a linear combination of biased and correlated estimates with application to forecasting.

The decision fusion problem may be seen as an M hypothesis detection problem [22] with hypotheses $h_m \in H$ where N individual detector decisions x_n form the observation $\mathbf{x}$ [1]:

$$\mathbf{x} = (x_0, \ldots, x_{N-1})^{\mathrm{T}} \in \mathbf{X} \tag{2.9}$$

where T denotes the transpose. The task can then be approached using classical decision theory [59]. As illustrated in Figure 2.3, the observation space $\mathbf{X}$ is split into disjoint regions X_i, so that given an observation $\mathbf{x}$, if $\mathbf{x} \in X_i$ then hypothesis h_i is chosen. The expected cost or *risk* R for the decision system is then :

$$R = \sum_{h_i \in H} \sum_{h_j \in H} P_j \lambda(h_i, h_j) \int_{X_i} p(\mathbf{x}|h_j) d\mathbf{x} \tag{2.10}$$

[1]The individual hypotheses may themselves be vectors, as is the case for Motion Estimation, and so by convention should be in bold font. However, for the sake of clarity they have been left in normal font.

where $\lambda(h_i, h_j)$ is the cost associated with choosing hypothesis h_i when h_j is true, and $P_j = P(h_j)$ is the *a priori* probability of hypothesis h_j. $p(\mathbf{x}|h_j)$ is the probability distribution of observing $\mathbf{x}$ given that h_j actually occurred. Rewriting Equation(2.10) and using:

$$\int_{\mathbf{X}} p(\mathbf{x}|h_j) = 1 \tag{2.11}$$

gives [59]:

$$R = \sum_{h_j \in H} P_j \lambda(h_j, h_j) + \sum_{h_i \in H} \int_{X_i} \sum_{h_j \in H - h_i} P_j(\lambda(h_i, h_j) - \lambda(h_j, h_j)) p(\mathbf{x}|h_j) d\mathbf{x} \tag{2.12}$$

The first sum is the fixed cost of the decision system and the second term is the cost dependent on the choice of decision region boundaries. $\lambda(h_j, h_j)$ is the cost associated with choosing the correct hypothesis, and is typically 0. The minimum of this risk function is when the decision regions X_i are assigned to minimise the integrands in (2.12) (assigning X_i to cover observation $\mathbf{x}$ is equivalent to choosing hypothesis h_i given $\mathbf{x}$), and so h is chosen if:

$$h = \arg\min_{h_i} \sum_{h_j \in H - h_i} P_j(\lambda(h_i, h_j) - \lambda(h_j, h_j)) p(\mathbf{x}|h_j) \tag{2.13}$$

Since the observation $\mathbf{x}$ is comprised of the decisions $x_0, \ldots, x_{N-1}$ with individual distributions $p(x_i|h)$, $p(\mathbf{x}|h)$ can be determined if the $p(x_i|h)$ distributions and their inter-dependencies are known.

In many fusion problems, the estimate $\hat{y}(\mathbf{x})$ is a value from a continuous range $y \in Y$ rather than a discrete hypothesis. This issue can be avoided by quantising the range into discrete hypotheses [47]. However, treating the problem as *estimation fusion* rather than decision fusion enables a more natural representation for certain applications. The risk associated with the decision system (2.10) may be re-formulated as follows [59]:

$$\begin{aligned} R &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \lambda(\hat{y}(\mathbf{x}), y)p(y, \mathbf{x})d\mathbf{x}dy & (2.14)\\ &= \int_{-\infty}^{\infty} p(\mathbf{x})\int_{-\infty}^{\infty} \lambda(\hat{y}(\mathbf{x}), y)p(y|\mathbf{x})dyd\mathbf{x} & (2.15)\end{aligned}$$

with R minimised when $\hat{y}(\mathbf{x})$ is assigned to minimise the inner integral.

2.4 Discussion

The choice of fusion strategy is typically application dependent. The previous sections have examined a number of Information Fusion applications reported in the literature and have identified a variety of fusion strategies from these examples. The application considered by this thesis is Motion Estimation, this section discusses the relative merits of the different fusion strategies for this task.

Since a key motivation for the work in this thesis is to take advantage of the differences in performance between Motion Estimation techniques under different conditions, it is important that the chosen fusion strategy should be able to take confidence measures into account. Simple committee voting methods do not take account of uncertainty in the individual decisions. Rank based methods go some way to dealing with this, but still do not allow for differences in the performance of individual techniques. Fuzzy, Dempster-Shafer and Probabilistic methods do enable confidence to be taken into account.

Classification fusion strategies such as committee methods, clustering methods and Dempster-Shafer evidence theory work with discrete hypotheses, however motion estimates are in general continuous values. The motion vectors could be quantised to enable discrete value methods to be applied. However as precision requirements increase, the number of hypotheses must increase accordingly and large numbers of hypotheses can be problematic for classification fusion strategies, especially as the difference between classes becomes negligible. Fuzzy Set and Bayesian theory, however, are both easily extended to the estimation fusion problem.

Fuzzy theory is sometimes reported to subsume probability theory and enable representations of uncertainties that can not be expressed by probabilities [97]. However this is a very contentious issue. Laviolette and Seaman [101] argue that Fuzzy Set Theory does not offer anything more than probability theory. Applying fuzzy set theory to a problem requires that fuzzy representations, operators and defuzzification operators be defined. It is not clear how this should be done and in the literature the choice is often subjective and application dependent [96].

The performance of Bayesian based fusion theory is typically reported favourably compared to other methods in the literature. Laviolette and Seaman [101] note that when comparisons are made in the literature, techniques based on probability theory are typically reported to equal or outperform fuzzy based techniques. Dasarathy reports preliminary results from a comparison of a variety of fusion strategies including Fuzzy and Bayesian based strategies applied to decision making for ballistic missile defence applications [102]. Although he draws no strong conclusions from these early results, the Bayesian based fusion strategy gives the best performance in terms of target ID accuracy and false alarm rates. Lee *et al.* [12] compare a statistical approach based on Bayesian theory against an evidential approach when applied to land classification from remote infrared and visible light sensors, they report similar performance for both techniques. Ben-Yacoub *et al.* [19] compare a variety of fusion strategies in their multi-modality personal identity verification approach and report that a Bayesian based strategy yields the best performance.

In summary, a fusion strategy for Motion Estimation should have some mechanism for representing uncertainty and should work well with a continuous valued domain and range. Committee based methods meet neither of these criteria. Clustering and Dempster-Shafer methods do take confidence measures into account, but are based on discrete hypotheses. Information Fusion based on Bayes' criterion extends readily to continuous values, and has been compared favourably to alternative techniques in the literature. Fuzzy Set theory can be used to represent continuous values, but the choice of fuzzification, aggregation and defuzzification operators is subjective and it is not clear that the use of Fuzzy Set theory leads to any advantage over probabilistic methods. Probabilistic methods have also been shown to to be very effective at representing motion in image sequences [103]. For these reasons, this thesis adopts a fusion strategy based on Bayes' criterion. The following section describes the Bayesian based approach in more detail.

2.5 Information Fusion with Bayes' Criterion

Bayesian estimation theory provides a popular framework for Information Fusion which is appropriate for application to fusing motion estimates. Section 2.3.5 gave a brief introduction to this approach. This section discusses the approach in greater detail.

2.5.1 Cost Functions

Three common cost functions are the Minimum Probability of Error (MPE), Absolute Difference (AD) and Squared Error (SE) functions. These are defined below:

$$\lambda_{\text{MPE}}(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{otherwise} \end{cases} \tag{2.16}$$

$$\lambda_{\text{AD}}(a,b) = |a-b| \tag{2.17}$$

$$\lambda_{\text{SE}}(a,b) = (a-b)^2 \tag{2.18}$$

The MPE Cost function penalises all errors equally, while the AD and SE costs penalise large errors more than small errors. All these cost functions have zero cost for a correct solution, which is a reasonable assumption for most problems.

The AD and SE functions require a difference operator, and so can not always be meaningfully applied to decision fusion. The MPE cost function, however, can be substituted in (2.13) to give:

$$h_i = \arg\min_{h_i} \sum_{h_j \in H - h_i} P_j p(\mathbf{x}|h_j) \tag{2.19}$$

$$= \arg\min_{h_i} \sum_{h_j \in H - h_i} p(\mathbf{x}) P(h_j|\mathbf{x}) \tag{2.20}$$

where (2.20) follows by application of Bayes' rule. This is an equivalent strategy to choosing the hypothesis with maximum a posteriori probability $P(h|\mathbf{x})$:

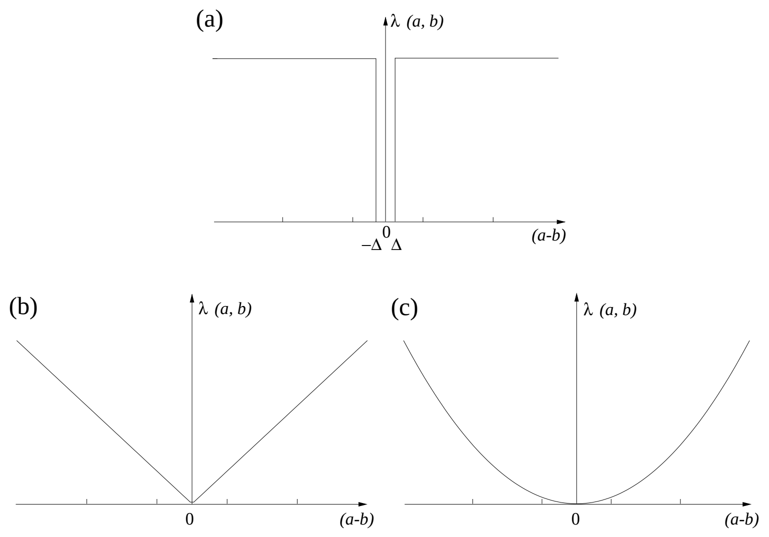


Figure 2.4: *Example cost functions a) the MPE cost function, b) the AD cost function and c) the SE cost function.*

$$h = \arg\max_h P(h|\mathbf{x}) \tag{2.21}$$

The MPE, AD and SE cost functions can all be meaningfully applied to Estimation Fusion (2.15). The MPE cost function is re-written as:

$$\lambda_{\text{MPE}}(a,b) = \begin{cases} 0 & \text{if } ||a-b|| < \Delta \\ 1 & \text{otherwise} \end{cases} \tag{2.22}$$

with Δ an application dependent value, but typically small. This function is illustrated along with the AD and SE cost functions in Figure 2.4. The MPE cost function can be substituted into (2.15) to give:

$$R = \int_{-\infty}^{\infty} p(\mathbf{x}) \left[1 - \int_{\hat{y}(\mathbf{x})-\Delta}^{\hat{y}(\mathbf{x})+\Delta} p(y|\mathbf{x}) dy \right] \tag{2.23}$$

which is clearly minimised where $\hat{y}(\mathbf{x})$ is chosen to maximise $\int_{\hat{y}(\mathbf{x})-\Delta}^{\hat{y}(\mathbf{x})+\Delta} p(y|\mathbf{x})dy$. As Δ approaches 0 this is equivalent to choosing $\hat{y}(\mathbf{x})$ to maximise the a posteriori distribution $p(y|\mathbf{x})$:

$$\hat{y}_{\text{MPE}} = \arg\max_{\hat{y}(\mathbf{x})} p(\hat{y}(\mathbf{x})|\mathbf{x}) \tag{2.24}$$

Assuming scalar estimates, the SE cost function can be substituted into the inner integral of (2.15) to give:

$$\hat{y}_{\text{SE}} = \arg\min_{\hat{y}(\mathbf{x})} \int_{-\infty}^{\infty} (\hat{y}(\mathbf{x}) - y)^2 p(y|\mathbf{x})dy \tag{2.25}$$

taking derivatives with respect to $\hat{y}(\mathbf{x})$ and setting the result to 0 gives [59]:

$$\begin{aligned} & \int_{-\infty}^{\infty} 2(\hat{y}_{\text{SE}}(\mathbf{x}) - y)p(y|\mathbf{x})dy = 0 && (2.26)\\ \Rightarrow\ & 2\hat{y}_{\text{SE}}(\mathbf{x}) \int_{-\infty}^{\infty} p(y|\mathbf{x})dy - \int_{-\infty}^{\infty} 2yp(y|\mathbf{x})dy = 0 && (2.27)\\ \Rightarrow\ & \hat{y}_{\text{SE}}(\mathbf{x}) = \int_{-\infty}^{\infty} yp(y|\mathbf{x})dy && (2.28) \end{aligned}$$

where (2.27) comes from using $\int_{-\infty}^{\infty} p(y|\mathbf{x})dy = 1$. This is just the average value of the distribution.

Similarly, substituting the AD cost function into the inner integral of (2.15) gives:

$$\begin{aligned} \hat{y}_{\text{AD}} &= \arg\min_{\hat{y}(\mathbf{x})} \int_{-\infty}^{\infty} |\hat{y}(\mathbf{x}) - y| p(y|\mathbf{x})dy && (2.29)\\ &= \arg\min_{\hat{y}(\mathbf{x})} \int_{-\infty}^{\hat{y}(\mathbf{x})} (\hat{y}(\mathbf{x}) - y)p(y|\mathbf{x})dy + \int_{\hat{y}(\mathbf{x})}^{\infty} (y - \hat{y}(\mathbf{x}))p(y|\mathbf{x})dy && (2.30) \end{aligned}$$

Again taking derivatives with respect to $\hat{y}(\mathbf{x})$ and setting the result to 0 gives [59]:

$$\int_{-\infty}^{\hat{y}(\mathbf{x})} p(y|\mathbf{x})dy = \int_{\hat{y}(\mathbf{x})}^{\infty} p(y|\mathbf{x})dy \tag{2.31}$$

which suggests that the fused estimate should be the median of the distribution.

2.5.2 Combining Distributions

The joint conditional distribution $p(\mathbf{x}|y)$ can be determined from the individual distributions $p(x_i|y)$ and their inter-dependencies. For example, if the individual estimates are statistically independent, then the joint probability distribution is just the product of the individual distributions:

$$p(\mathbf{x}|y) = \prod_{k=0}^{N-1} p(x_k|y) \tag{2.32}$$

From Equation(2.24) it was seen that the MPE cost function leads to a fusion strategy that chooses $\hat{y}(\mathbf{x})$ to maximise the a posteriori distribution $p(y|\mathbf{x})$. By applying Bayes' rule and substituting (2.32), $p(y|\mathbf{x})$ can be expressed as follows:

$$p(y|\mathbf{x}) = \frac{p(y)\prod_{k=0}^{N-1} p(x_k|y)}{p(\mathbf{x})} \tag{2.33}$$

substituting into 2.24 yields:

$$\hat{y}(\mathbf{x}) = \arg\max_{y} p(y) \prod_{k=0}^{N-1} p(x_k|y) \tag{2.34}$$

with $p(\mathbf{x})$ removed because it is the same for all $\hat{y}$. This can be written in terms of the posterior distributions from the individual estimators as follows:

$$\hat{y}(\mathbf{x}) = \arg\max_{y} p(y)^{-(N-1)} \prod_{k=0}^{N-1} p(y|x_k) \tag{2.35}$$

This was reported in its discrete form, by Kittler *et al.* [84, 85, 100] as follows:

$$h = \arg\max_{h_i} P_i^{-(N-1)} \prod_{k=0}^{N-1} P(h_i|x_k) \tag{2.36}$$

This *product rule* for classifier combination is severe in that if any classifier allocates a very low or zero probability to a hypothesis then the risk associated with that hypothesis will be high, even if the remaining classifiers allocate the hypothesis high probabilities.

Kittler *et al.* [84, 100] suggest that it be further assumed that the individual posterior probabilities are not significantly different from the prior probabilities, this is formulated as follows:

$$p(y|x_k) = p(y)(1 + \delta_{y,x_k}) \tag{2.37}$$

where δ_{y,x_j} is very small. (2.35) can now be written:

$$\hat{y}(\mathbf{x}) = \arg\max_{y} p(y) \prod_{k=0}^{N-1} (1 + \delta_{y,x_k}) \tag{2.38}$$

by expanding the product and ignoring the second order and higher terms, they derive a less severe combination strategy based on summing the individual distributions:

$$\hat{y}(\mathbf{x}) = \arg\max_{y}(1 - R)p(y) + \sum_{k=0}^{N-1} p(y|x_k) \tag{2.39}$$

Kittler *et al.* [84, 100] show that this *sum rule* is less sensitive to errors than the product rule for combining classifiers. Similarly, in the statistical literature averaging the individual distributions has become a popular approach to combining models [104].

The independence assumption is not generally valid, and so will at best lead to a loss of information in the fusion process and possibly to classification errors. Anandaligam and Chen [26] describe how biased and correlated distributions may be combined for Information Fusion. Typically, difficulties in determining inter-class dependencies force this assumption. However, Lee *et al.* [12] note that the assumption also has some benefits for information fusion, as information sources are treated separately so new sources can easily be added and the loss of a source is easily dealt with.

2.5.3 Comment on Fuzzy Approaches

An important problem when Fuzzy approaches are adopted for Information Fusion is how to choose appropriate fuzzification, aggregation and defuzzification operators. Although, for reasons discussed in Section 2.4, a Fuzzy approach has not been adopted for this thesis, the author notes that an analogy with the Bayesian approach discussed in this section can help in the the choice of these functions.

The *a-posteriori* probabilities $p(y|x_i)$ represent uncertainty in the individual estimates and so are analogous to fuzzy membership functions, which suggests that the fuzzifying function should describe the likelihood of an observation with respect to different events. The combination of these to give $p(y|\mathbf{x})$ is analogous to applying the fuzzy aggregation functions and so the inter-dependencies of solutions should be modelled by the choice of aggregation functions. Finally, the defuzzifying function is analogous to solving Equation(2.14), which suggests that the defuzzifying function should model both *a priori* knowledge and risk. This analogy has been reported in [47] and [48].

2.6 Summary

This chapter has introduced the Information Fusion research field. Information Fusion has a wide range of applications and there exist a variety of approaches which are to an extent application dependent. Popular fusion strategies include Committee based methods, clustering methods, techniques based on Fuzzy set theory, Dempster-Shafer evidence theory and Bayesian decision theory. Of these, Bayesian based decision theory was found to be the most appropriate strategy for combining motion estimates as confidence measures can be taken into account and the techniques adapt well to a continuous valued range and domain.

Chapter 3
Analysis of Image Motion

3.1 Introduction

The problem of detecting and interpreting motion from image sequences is important for a number of applications including image sequence compression [29, 30], robot control [36], navigation [35] and tracking [37]. A survey of the field is given by Mitiche and Bouthemy in [105]. Beauchemin and Barron [106] provide a survey of Motion Estimation in which they classify techniques as Differential, Correlation or Frequency based. This work was later extended in a technical report by Barron *et al.* [107]. This chapter presents an overview of motion related research in image processing, and presents three examples of motion techniques which are used in this thesis as examples for Information Fusion.

When the image of a scene changes there are a number of possible causes, including altered illumination, characteristics of the imaging system, noise, and the motion of objects in the real world relative to the imaging device. Two prominent areas of research looking at imaged scene changes are *Motion Estimation* and *Motion Compensation*. Motion Compensation is part of the image compression field, the compression benefit comes from the removal of temporal redundancy in frame sequences. Motion Estimation research is interested in determining when imaged scene changes are caused by object motion, and in determining what real world motion caused the changes. A simpler version of Motion Estimation is *Motion Detection*, where techniques determine when image scene changes are caused by real world motion but do not try to determine what motion occurred. Figure 3.1 illustrates this decomposition of the research field.

Although the theme of this thesis is Motion Estimation rather than Motion Compensation, many Motion Estimation techniques are derived from the Motion Compensation field and so it makes sense to consider techniques according to this taxonomy, which is the basis for the structure of this chapter: Section 3.2 and Section 3.3 give overviews of the Motion Compensation and Motion Estimation fields respectively. Section 3.4 presents the example techniques used in this thesis and discusses issues of timeliness in terms of the Accuracy-Efficiency trade-off problem in Motion Estimation.

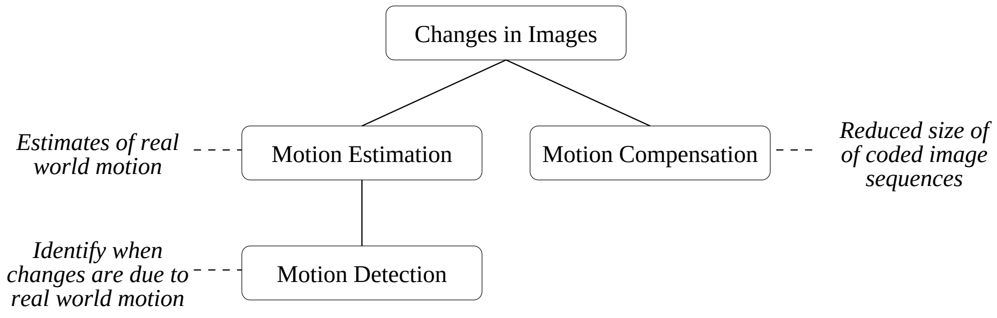

Changes in Images
Estimates of real world motion
Motion Estimation
Motion Compensation
Reduced size of of coded image sequences
Identify when changes are due to real world motion
Motion Detection

Figure 3.1: *A simple taxonomy for research on imaged scene changes.*

3.2 Motion Compensation

Typically, much of the scene in an image sequence remains constant over time. Also, the appearance of the moving parts in the image tends to vary slowly. This leads to temporal (or *inter frame*) redundancy in coded image sequences which a number of compression algorithms such as MPEG-1 [31] and CCITT H.261 [32] try to utilise. Techniques for Motion Compensation look for ways to predict the pel values in one frame from pel intensities and movement detected in other frames [29].

The effectiveness of a Motion Compensation technique is measured by the compression gain it enables and not by how well the technique estimates motion in the real world. This can be evaluated either as a function of prediction error or from the entropy of the encoded signal [108]. Motion vectors which accurately describe real world motion do not necessarily lead to the best compression benefits. For example, if a large object of constant intensity is moving in an image sequence, then setting the motion vectors inside the object boundary to be zero can give as good compression benefits as the true motion vectors, even though the image is moving. This is because the zero motion vectors are sufficient to enable a good reconstruction of the image.

Most Motion Compensation techniques can be classified as one of two prominent strategies: *Correlation Based* and *Pel-Recursive* methods. A third strategy involves working in the *Frequency Domain*. This section discusses these different approaches.

3.2.1 Correlation Based Methods

If it is assumed that local pel motion is uniform, then pels may be grouped into zones and their motion considered together. The motion of a zone may be estimated by searching for the most similar zone, according to some measure of similarity, in subsequent frames. Correlation based methods differ mainly in their choices of similarity measure, zone shape and in the search algorithm used to find the most similar zone. The choice of search algorithm mainly affects speed while zone shape and the similarity measure more directly affect accuracy.

Similarity Measures

One simple similarity measure D is the sum of some pel by pel comparison function f over two isomorphic zones. For square zones of width N pels this is formulated as follows:

$$D = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(I(x, y, t), I(x + u, y + v, t + \Delta t)) \tag{3.1}$$

where $I(x, y, t)$ is the intensity of the pel $(x, y)^T$ at time t. Different motion vectors $(u, v)^T$ are examined to find the vector that yields the greatest similarity to the reference block. This rectangular block based approach is commonly termed the Block Matching Algorithm (BMA).

Common comparison functions are the AD (Absolute Difference) where $f(a, b) = |a - b|$, the SD (Squared Difference) where $f(a, b) = (a - b)^2$, and the CC (Correlation Coefficient) where $f(a, b) = ab$. With the AD and SD functions, low values of D suggest greater similarity, and so it is more appropriate to call these measures of *distortion*. The CC measure has a maximum where the zones are most similar.

Zone Shape and Size

Although by far the most common zone shape in motion compression is a square block of pels, usually 8x8 or 16x16, a number of alternatives have been used. The generalised transformation of Seferidis and Ghanbari [109, 110], for example, allows blocks to be any convex quadrilateral, and allows the shape of these quadrilaterals to change over time. This enables a more accurate portrayal of the motion in the scene, but there is a complexity trade-off as the search algorithm must now find the best parameters for the transformation as well the translation.

The size of the block affects the spatial scale at which motion may be detected [108, 110–112]. A number of schemes take account of this and adopt variable block sizes over the image. There is also a trade off between sensitivity to noise with smaller blocks and covering different regions with larger blocks. Kanade and Okutomi [111] describe a method for adaptively selecting the block size based on local intensity variation and disparity in the context of Stereo Matching. Another approach to selecting block size is to start with large blocks and segment them according to some splitting criterion. Typically, a quad-tree segmentation is used because this offers a good trade-off between accurate segmentation and representational complexity[110]. A number of criteria have been suggested for selecting appropriate block sizes, these are typically based either on the prediction error or on a measure of the entropy of the coded signal[108].

Li and Lin [113] use the prediction error in their multi-resolution block matching strategy. If the prediction error is above some threshold then the block is split. Seferidis and Ghanbari [110] define the *Absolute Temporal Difference* splitting criterion as the sum of absolute differences between in corresponding blocks in subsequent frames. If this difference is above some threshold, then the block is split.

Malo *et al.* [108] note that such criteria do not take account of the encoding method. Compression algorithms such as MPEG-1 [31] transform blocks into the frequency domain and transmit the quantised coefficients, as a result the prediction error does not necessarily reflect the compression performance. They use a variable block size chosen using a criterion based on the spectral entropy of the frame difference. Entropy based methods are more directly related to effectiveness of compression approaches than prediction error methods and so are more appropriate for Motion Compensation.

Search Algorithms

The simplest search algorithm is to compare the pel zone in the current frame at time t with all potential locations of the zone in the frame at time $t + \Delta t$. Typically, it is assumed that there is some maximum velocity that needs be considered, and so the search radius w can be restricted [29]. The number of potential locations L which must be considered is then:

$$L = (w + 1)^2 \tag{3.2}$$

as illustrated in Figure 3.2. This search algorithm is known as the Full or Exhaustive Search Algorithm (FSA). The FSA is computationally expensive and so too slow for many applica-

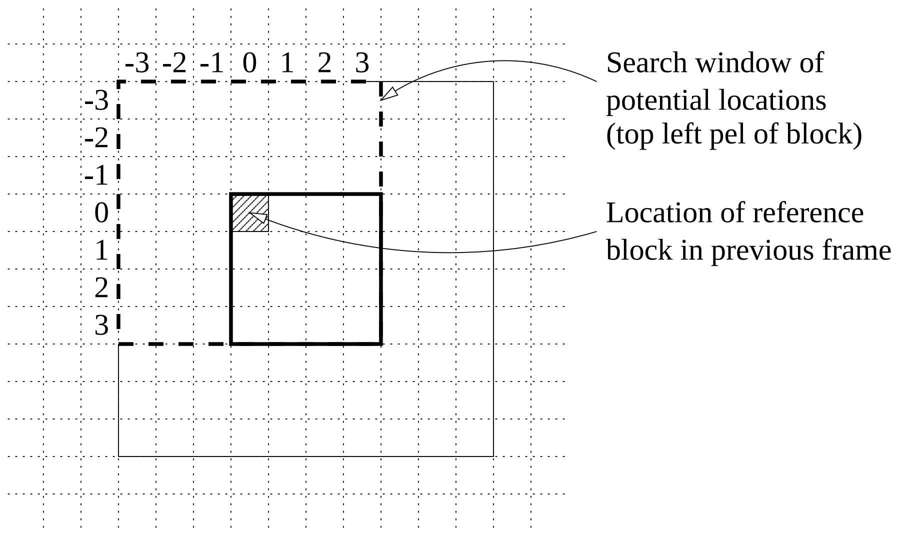

-3 -2 -1 0 1 2 3
-3
-2
-1
0
1
2
3
Search window of potential locations (top left pel of block)
Location of reference block in previous frame

Figure 3.2: *Example search window $w = 3$ for a 4×4 pel block with the block location indexed by the top left corner.*

tions, and so there has been a great deal of attention given to find faster search algorithms. A common theme is to find ways to eliminate unlikely matches from the search, so reducing the required computation. An interesting consequence of such efficient implementations is that the run time of the methods can become dependent on the image sequence. Alternatively, methods of reducing the complexity of (3.1) may be sought.

One way to speed block matching is to find faster implementations which achieve equivalent accuracy but at lower computational cost. Examples of such fast implementations include the Successive Elimination Algorithm (SEA) proposed by Li and Salari [114] and the Block Sum Pyramid algorithm (BSPA) proposed by Lee and Chan [115]. Both these approaches identify functions of pel intensities which are easier to compute than the full similarity measure but which can be used to reject many blocks as being very dissimilar, so saving time calculating the full similarity measure for those blocks.

Some fast block matching algorithms accept a loss of accuracy for improved computational efficiency. A popular method is to assume that the distortion function used to compare blocks increases monotonically as the search moves away from the best match position[116]. Since many real images will not be consistent with this monotonic distortion assumption, a level of error is introduced into such algorithms. Figure 3.3 shows an example plot of the AD distortion measure for a 5×5 reference block with a 5 pel search range. In addition to the global minimum,

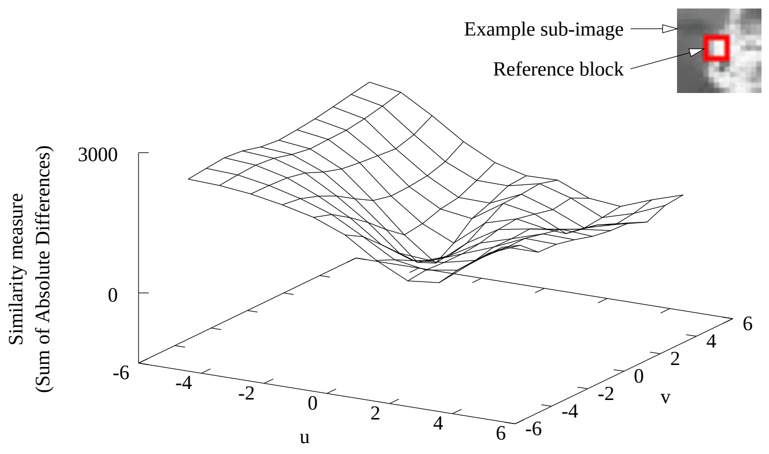


Figure 3.3: *Example distortion surface using the AD as the difference measure for a 5×5 reference block and a search range of 5 pels.*

there is a local minimum which may cause errors in search algorithms that make the monotonic distortion assumption.

The most popular fast search approach that takes advantage of the monotonic distortion assumption is to use a set of search patterns which progressively refine the motion estimate [117]. The Three-Step-Search (TSS) [118], which uses a fixed pattern of search locations distributed over the search window, has become one of the most well known of these algorithms. Li *et al.* [119] note that the distribution of motion vectors is typically biased towards no motion and so present a centre-biased version of the TSS which gives faster performance on images with small motion vectors, they call this the New TSS (NTSS). The NTSS considers more motion vectors than the TSS, but uses an early stopping criterion to save searching unlikely locations. Po and Ma [120] present the Four-Step-Search FSS algorithm which uses concepts from the TSS and NTSS to give similar (though slightly worse) performance to the NTSS at lower computational cost. Nisar and Choi [121] recently presented a centre-biased search algorithm which applies the assumption of monotonically increasing distortion to reduce the number of motion vectors considered. They present results which show both error and speed improvements over the TSS algorithm.

The 1D Full Search algorithm proposed by Chen *et al* [122] is also an approximate fast versions of the FSA which make the monotonic distortion assumption. Rather than use search patterns to minimise the distortion measure, this approach uses successive 1D searches in alternating horizontal/vertical directions, starting each search at from the pel location with the best match value from the previous step. The search length is halved every horizontal iteration, until a single pel is identified.

An alternative approximation approach to speeding block matching is to estimate the general direction of block motion using some very fast technique and then restrict the search to locations in that direction. Young and Kingsbury [45] propose that their frequency-domain based Motion Estimation method should be used to direct the search for optimal motion vectors, and that final selection of the best location be implemented using the MSE distortion function.

Feng *et al.* [123] define the *Bitplane Matching Criterion*, in which bit planes are associated with blocks so that a pel's bit plane element is 0 if the pel value is less than the block mean, else 1. Bit planes which are very dissimilar to the reference block are unlikely to provide good matches so are discarded from the search. In general, the computational savings made by rejecting unlikely matches are far greater than the small additional cost of calculating and comparing the bit planes.

A number of techniques use knowledge of the compression problem to improve on the FSA speed. Such approaches can lead to Motion Compensation techniques which yield good compression benefits but often give motion estimates which do not correspond to real world motion. For example, Coban and Mersereau [124] describe a fast implementation of the FSA for Motion Compensation applications where the bit rate for the compressed image sequence is constrained. The rate constraint can be used to reject potential motion vectors which would adversely affect the bit rate, even if those vectors accurately portray the real world motion in the scene.

Fan and Gan [125] note that there are some cases where motion compensation can not improve the compression rate, for example where much of the interframe difference is due to new objects being uncovered. By identifying such cases using simple correlation methods, the time expensive FSA can be avoided altogether.

Camus [1] notes that if the search area and block width are fixed but the number of frames t is allowed to vary, then the computational complexity of the search is linear in t rather than quadratic in N. This enables sub pel accuracy in the motion vectors. However, in real terms this mathematical trick gives no improvement to FSA run-times.

3.2.2 Pel-Recursive

In the mid 1970's, Limb and Murphy [126, 127] developed a very simple technique for estimating the motion of a single translating object based on the observation that the magnitude of the Frame Difference Signal (FDS) (or temporal intensity gradient) increases linearly with speed at low speeds. They showed how to calculate the horizontal and vertical components of the motion vectors from a function of the FDS and Element Difference Signal (EDS) (or spatial intensity gradient). The method developed by Limb and Murphy was later shown to be a special case of the *Pel-Recursive* methods[30, 128].

Pel-Recursive techniques start with the assumption that pel motion over moving areas is translatory. It follows that:

$$I(x, y, t) = I(x - u, y - v, t - \Delta t) \tag{3.3}$$

where again $I(x, y, t)$ is the intensity of the pel $(x, y)^T$ at time t and $(u, v)^T$ represents the motion of the pel. The FDS can then be calculated as follows:

$$\begin{aligned} FDS(x, y, t) &= I(x, y, t) - I(x, y, t - \Delta t) && (3.4)\\ &= I(x, y, t) - I(x + u, y + v, t) && (3.5)\\ &\simeq u\frac{\partial I}{\partial x} + v\frac{\partial I}{\partial y} && (3.6)\\ &= \mathbf{u}\nabla I && (3.7) \end{aligned}$$

where (3.6) comes from expanding the last term as a Taylor series about (x, y) and ignoring the higher order terms. This is known as the *Motion Gradient Constraint*. ∇I is the spatial gradient of I, and $\mathbf{u} = (u, v)$ is the motion vector. This can be solved using linear regression to give [30]:

$$\hat{\mathbf{u}} = -\left[\sum \nabla I(x, y, t) \cdot \nabla I(x, y, t)\right]^{-1} \left[\sum FDS(x, y) \cdot \nabla I(x, y, t)\right] \tag{3.8}$$

where the sums are over the moving area. This can be used to determine the appropriate motion vector for the pels in the moving area.

Pel-Recursive methods form the basis of the Motion Estimation Differential methods, which will be discussed in more detail later in this chapter.

3.2.3 Working in the Frequency Domain

In a number of popular compression algorithms such as MPEG-1 [31], images are converted to the Discrete Cosine Transform (DCT) domain. As a result, a lot of hardware has been developed for this conversion, which in turn has lead to interest in estimating motion in the frequency domain [129]. The 2D DCT is defined for an $N \times N$ block of pels as follows [130]:

$$\text{DCT}(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x,y) \cos\left(\frac{(2x+1)u\pi}{2N}\right) \cos\left(\frac{(2y+1)v\pi}{2N}\right) \tag{3.9}$$

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{iff } u = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases} \tag{3.10}$$

The DCT$(0,0)$ coefficient is just the average intensity of the image block and is commonly termed the DC coefficient. The remaining coefficients, termed the AC coefficients, may be interpreted as $N \times N$ convolution masks (or basis functions) with the horizontal and vertical spatial frequencies increasing with i and j. The masks for an 8×8 DCT are illustrated in Figure 3.4, where the coefficients have been rescaled to lie in the range $[0, 255]$ and interpreted as greyscale images.

If a feature moves from one point to another inside the window of a frequency domain transformation then the transformed signal will incorporate this shift. Koc and Liu [42–44] use this shift property of the discrete Cosine and Sine transforms [131] to detect the movement of high intensity pels on a zero intensity background. By applying an edge detector to the original image, this technique can be used to determine the motion of the edges.

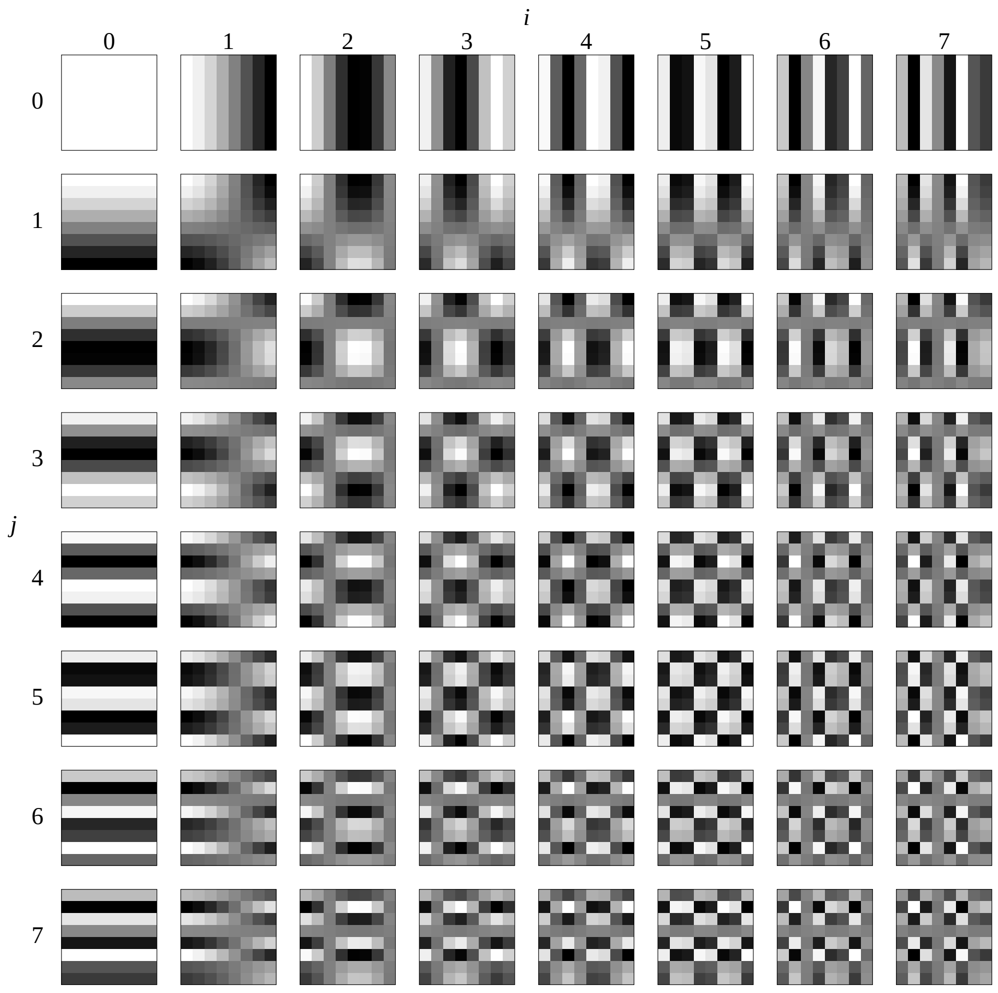


Figure 3.4: *DCT basis functions for an* 8×8 *transform.*

3.2.3.1 Motion Direction Estimates from Differenced DCT Images

The pixel differences between subsequent frames typically have large magnitudes at the edges of moving objects. For example, a rectangular object moving horizontally through an imaged scene will yield difference frames containing vertical high magnitude bars. Motion can be estimated by detecting and classifying such features. In this section a motion direction estimation strategy developed by the author (and reported in [53]) that operates in the DCT domain is described.

As the AC coefficients can be interpreted as convolution masks, they may be used to detect spatial features in the images. Kim and D.Lee [132] used the signs of DCT AC coefficients to distinguish between edge patterns in their Vector Quantisation technique. Lee and Crebbin[133] showed that the DCT AC coefficients, normalised by one of the coefficients, can be used to detect edge features independently from the pixel intensity values. Pagliari and Dennis [134] use DCT coefficients to classify edge features in their disparity estimation algorithm for stereo vision.

As in the BMA, it is assumed that frame differences are due to the translatory movement of an object, with respect to the imaging device, in front of a still background under constant and uniform lighting[116]. A high intensity object moving horizontally in front of a darker background will have a bar of positive values at the leading object edge and negative values at the following edge. It is desirable that the feature classification should be independent of the relative intensities of the background and object, this can be achieved by normalising the AC coefficients using the DC coefficient.

Feature classification is achieved by pattern matching $M < N^2$ of the DCT coefficients against F feature models. The choice of features is application dependent, Lee and Crebbin [133] used 24 edge models while Pagliari and Dennis [134] found that 8 models were sufficient. The results presented in this section use 8 model features from a 4×4 DCT and 6 coefficients $(u, v) = \{(0,1),(0,2),(0,3),(1,0),(2,0),(3,0)\}$. The model features are illustrated in Figure 3.5 and correspond to the edge features that would be expected from objects moving in (a,b) horizontal, (c,d) vertical, (e,f) $135°$ and (g,h) $45°$ directions.

Previous authors have only applied the DCT feature classification to areas of high detail [133, 134]. Similarly, in the differenced images considered here areas of low detail correspond to no-motion areas. The magnitude of the DC coefficient can be used to check that there is sufficient

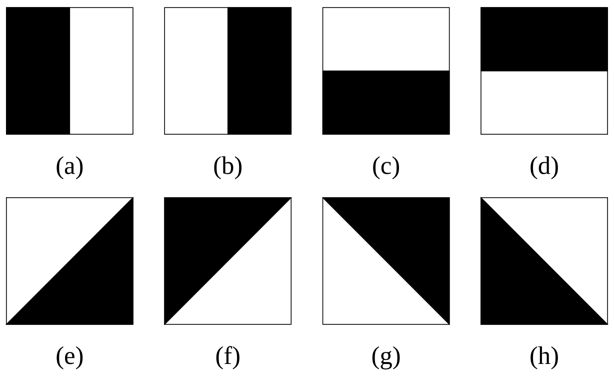


Figure 3.5: *Model Edge Features for (a,b) Horizontal, (c,d) Vertical, (e,f) $135°$ and (g,h) $45°$ motion.*

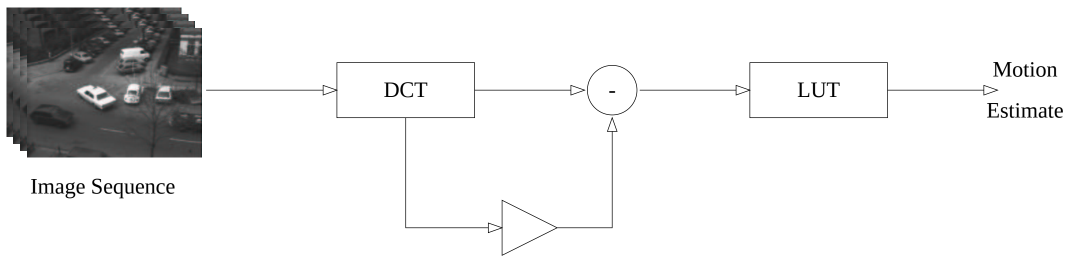


Figure 3.6: *Block diagram illustrating an technique for estimating motion direction directly from differenced DCT images. The Look Up Table (LUT) comprises 8 model edge features which are illustrated in Figure 3.5.*

difference between the two frames to suggest motion has occurred.

It can be readily verified that for any two pixel blocks A and B:

$$\text{DCT}(A - B) = \text{DCT}(A) - \text{DCT}(B) \tag{3.11}$$

Given this result, there is no need to know the original intensity values when creating the differenced DCT coefficients, and so all processing may be done in the DCT domain. Figure 3.6 shows a block diagram that illustrates the technique.

Figure 3.7(a) shows an example frame from an experimental image sequence in which a rectangular test card with a checkered pattern moves at constant speed horizontally from right to left across the field of view. Figure 3.7(b) shows results from detecting motion using a 4x4 DCT to

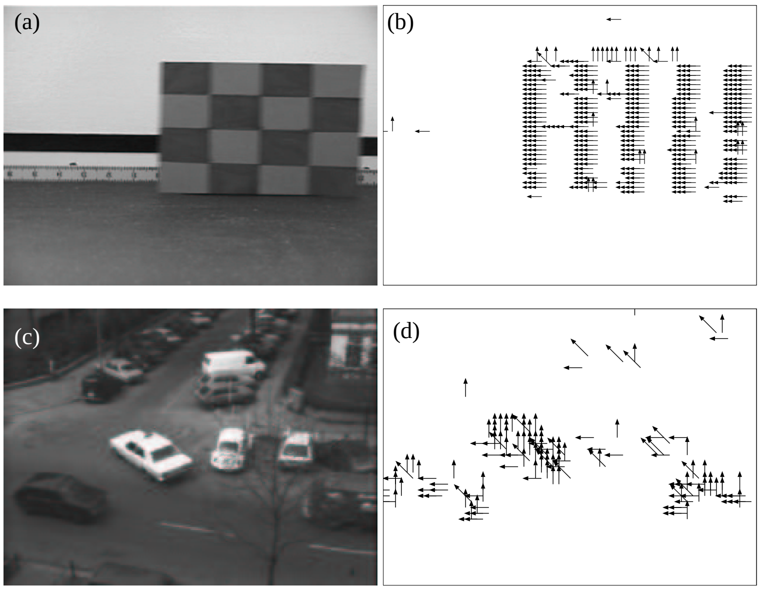


Figure 3.7: *Results showing (a) an example frame from the horizontal tracked sequence, (b) the estimated motion at this frame, (c) an example frame from the Hamburg Taxi sequence, (d) the estimated motion at this frame.*

classify the frame differences as arising from horizontal, vertical, $45°$ or $135°$ motion. Arrow heads are drawn to make the motion vectors clearer, however the vectors should be interpreted as bi-directional. The technique has identified the horizontal motion of the object. Figure 3.7(c) shows an example frame from the well known Hamburg Taxi sequence [135], Figure 3.7(d) shows a typical result from applying the motion estimation technique to this sequence. The technique has detected and appropriately classified the motion of the three vehicles.

Clearly the motion estimates from this strategy are not very precise and contain no magnitude information. It is proposed that this technique might have application in fast BMA search algorithms for Motion Compensation by giving a very fast though rough estimate of the general motion direction which can be used to restrict the search space. In particular, the technique might be used with compression standards such as MPEG-1 where the DCT coefficients are easily available.

3.3 Motion Estimation

The theme of this thesis is the application of Information Fusion to Motion Estimation rather than Motion Compensation. The purpose of Motion Estimation techniques is to determine when changes in the image of a real world scene are caused by the motion of objects relative to the imaging device, and to determine what real world motion caused those changes. Although historically Motion Estimation techniques are often derived directly from Motion Compensation approaches, the underlying aims and paradigms of the two fields are very different and so lead to important differences in the techniques.

To measure the effectiveness of a motion estimation technique, the estimated motion fields for an image sequence must be compared with the known real world motion for that scene. It is difficult to create a real data set for which ground truth motion is known, and so comparisons tend to be qualitative or based on artificially generated data sequences, such as the Diverging and Translating Tree sequences[107, 136]. However, the performance of a technique on artificial data only shows how the techniques perform under ideal conditions.

As with Motion Compensation, most Motion Estimation techniques can be classified as either Correlation or Pel-Recursive methods. In Motion Estimation these are usually termed *Feature Correspondence* or *Differential* methods respectively. This section discusses these approaches.

3.3.1 Feature Correspondence

Feature correspondence methods search for corresponding features between one frame and successive frames. Methods differ mainly in their choice of feature. The choice of feature affects the search algorithms used to find corresponding features and the functions used to decide correspondence.

Zone Correspondence The zone correspondence approach to Motion Estimation is a special case of feature correspondence, where the feature being matched is the pel intensity pattern of the zone. This technique is directly derived from the Correlation based approach to Motion Compensation described in Section 3.2.1.

The SD and AD zone similarity measures are sensitive to uniform changes in intensity, this is useful for image sequence compression where such intensity changes are of interest, but can be problematic for Motion Estimation where real world motion and hence intensity structure is more important. Similarly, the CC measure is sensitive to changes in intensity magnitude, and the following normalised form of the Correlation Coefficient (NCC) is generally preferred [130, 137]:

100	150	154
99	155	156
101	153	157

(a)

1	3	5
0	6	7
2	4	8

(b)

Figure 3.8: *(a) Intensities of an example 3×3 pel block and (b) the corresponding rank matrix.*

$$\mathrm{NCC} = \frac{\sum(I(\mathbf{x},t) - \bar{I}_t)(I(\mathbf{x}+\mathbf{d},t+\Delta t) - \bar{I}_{t+\Delta t})}{\sqrt{(\sum(I(\mathbf{x},t) - \bar{I}_t)^2 \sum(I(\mathbf{x}+\mathbf{d},t+\Delta t) - \bar{I}_{t+\Delta t})^2)}} \tag{3.12}$$

where the sums are over the pels $\mathbf{x} = (x, y)$ in the zone, $\bar{I}_t$ is mean intensity for the zone at time t and $\mathbf{d} = (u, v)^T$ is the displacement. This measure has the advantage of being independent of scale changes in the intensities of the block pels.

An alternative class of similarity measures are the ordinal measures, which are based on determining differences between the rank matrices of compared blocks. Figure 3.8 shows an example 3×3 pixel block and it's corresponding rank matrix. A key advantage of difference measures based on the rank matrices is that changes to single pels, which might significantly affect measures such as the NCC, may have little or no change in the rank matrix. Bhat and Nayar [138] showed that ordinal measures outperform both SD and NCC measures in the context of stereo matching (a related problem) with salt and pepper noise. However, in Gaussian noise their rank based approach performs significantly worse than the SD and NCC measures.

The fast algorithms discussed in Section 3.2.1 can also be applied to Motion Estimation. However, some assumptions which were appropriate for Motion Compensation can lead to performance degradation in Motion Estimation. Typically, for example, the distortion measure does not increase monotonically from the global minimum, indeed the distortion surface can be multi-modal. In such cases fast search algorithms can easily fall into local minima, leading to errors in the motion estimates. Decroos *et al.* [34] describe a fast block matching search algorithm intended for surveillance applications. Their algorithm introduces a new checking point pattern which includes points between local minima rather than just looking at the neighbourhood of single local minimum at each step.

Region Correspondence Segmented images produce regions which can be used as features for correspondence. These are regions with respect to the imaged scene, and not in the sense of the regular pel zones from Zone Correspondence methods. Kalivas and Sawchuk [38] use the region formed by the central projection of an object as a feature for feature correspondence. They model motion in terms of affine transformations of these zones. Their algorithm assumes pre-segmented images.

As only a finite number of match positions are searched, motion estimates from region correspondence methods are typically discrete. For a technique that applies block matching to estimate the motion of a block of pixels in the next frame, the precision of the estimate will be limited to 1 pel/frame. Camus [1] notes that if the search is over t subsequent frames then the precision of the estimate will increase to $1/t$ pel/frame.

Camus' technique is a generalisation of correlation techniques using fixed block shape and size. As this technique is very illustrative of the correlation based motion estimating methods [1], it is used in this thesis to test the Information Fusion approaches when it is refered to as CAMUS.

3.3.2 Tracking with Zone Correspondence

The Block Matching Algorithm (BMA) is a popular correlation-based approach to motion estimation [116] and tracking [37, 39, 40]. In this approach, the motion of a block of pels, termed a *Reference Block*, is estimated by looking for the most similar block of pels in subsequent frames. Since the appearance of tracked objects can change over time, the reference block must be updated to take account of these changes.

3.3.3 Differential Approaches

Differential approaches to Motion Estimation are derived from Pel-Recursive work in the Motion Compensation field, and start with the assumption that the intensity of an imaged point is constant over time and over the motion of the real point. This may be formulated as follows [2]:

$$\frac{dI}{dt} = 0 \tag{3.13}$$

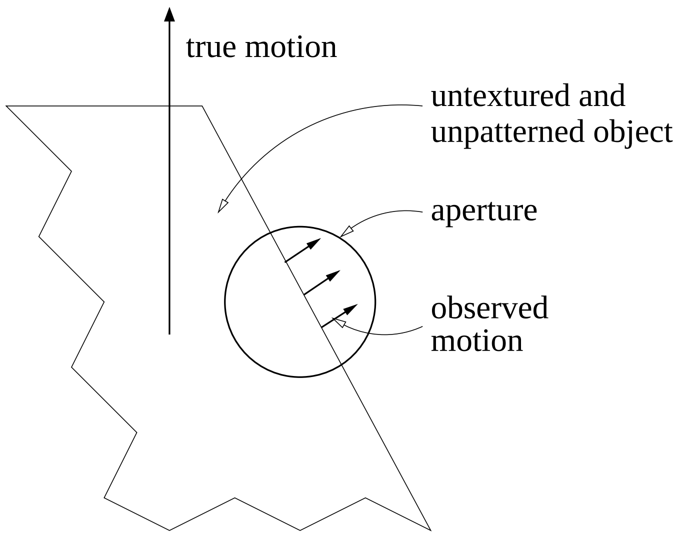


Figure 3.9: *Illustration of the aperture problem. Where a local area of the edge is observed, only the motion component perpendicular to the edge can be known.*

Expanding the left hand side gives:

$$E_M = \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} \tag{3.14}$$

where $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ are the horizontal and vertical motion components of the motion vector $(u, v)^T$ respectively. This is known as the *Motion Gradient Constraint* (E_M), or the *Optical Flow Constraint* [105, 106, 139, 140]. (3.14) is identical to (3.7) used in the Motion Compensation Pel-Recursion approach, with FDS $= \frac{\partial I}{\partial t}$.

(3.14) is the equation of a line in velocity space. The problem of solving (3.14) is ill-posed in the sense that the solutions are constrained to this line, not to a single point. This is commonly known as the *Aperture Problem* [2] and has a simple physical interpretation that if only a local area of an edge can be observed then only the motion component perpendicular to the edge can be known. This is illustrated in Figure 3.9. Additional regularisation constraints are required to determine a motion vector for a point [41]. Differential techniques differ mainly in their choice of additional constraints.

Horn and Schunck [2], who were one of the first to report the application of Pel-Recursive algorithms from the Motion Compensation field to Motion Estimation, proposed the *Smoothness*

Constraint (E_S), which assumes that motion varies slowly over the image. This is formulated as follows:

$$E_S = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{3.15}$$

The motion field is then determined by minimising the following error function over the image:

$$E = E_M^2 + \lambda E_S \tag{3.16}$$

where λ is chosen to weight the error in the Motion Gradient Equation with respect to the smoothness of the flow. This has the following multi-frame iterative solution (a discrete derivation for these equations is given in Appendix B.1):

$$u_{kl}^{n+1} = \bar{u}_{kl}^{n} - \frac{I_{x_{kl}}(I_x \bar{u}_{kl}^{n} + I_{y_{kl}} \bar{v}_{kl}^{n} + I_{t_{kl}})}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{3.17}$$

$$v_{kl}^{n+1} = \bar{v}_{kl}^{n} - \frac{I_{y_{kl}}(I_x \bar{u}_{kl}^{n} + I_{y_{kl}} \bar{v}_{kl}^{n} + I_{t_{kl}})}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{3.18}$$

where $I_{x_{kl}}$, $I_{y_{kl}}$ and $I_{t_{kl}}$ are the intensity derivatives with respect to x, y and t respectively at pel location (k, l). u_{kl} and v_{kl} are the horizontal and vertical motion components at pel location (k, l). $\bar{u}_{kl}$ and $\bar{v}_{kl}$ are the average horizontal and vertical motion components around the pel location (k, l).

Since this can take some time to converge, Horn and Schunck propose that it is allowed to run just once per frame. Over sufficient frames, a good approximation to the flow will be found. As this technique is a well known and popular illustration of the differential based motion estimating methods [105–107, 136], it is the second technique used in this thesis to test the Information Fusion approaches and is refered to as HORN.

A number of techniques tackle the Aperture Problem by assuming that the second order derivatives of the Motion Gradient Constraint (3.14) are also constant. Taking the second order derivatives yields the following system of equations:

$$\begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \frac{\partial^2 I}{\partial x \partial t} \\ \frac{\partial^2 I}{\partial y \partial t} \end{bmatrix} \tag{3.19}$$

the matrix on the LHS is the Hessian **H** of I. There exists a unique solution for u and v so long as **H** has an inverse:

$$u = \left(\frac{\partial^2 I}{\partial x^2} \frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial x \partial y} \right)^2 \right)^{-1} \left(\frac{\partial^2 I}{\partial y^2} \frac{\partial^2 I}{\partial x \partial t} - \frac{\partial^2 I}{\partial x \partial y} \frac{\partial^2 I}{\partial y \partial t} \right) \tag{3.20}$$

$$v = \left(\frac{\partial^2 I}{\partial x^2} \frac{\partial^2 I}{\partial y^2} - \left(\frac{\partial^2 I}{\partial x \partial y} \right)^2 \right)^{-1} \left(- \frac{\partial^2 I}{\partial x \partial y} \frac{\partial^2 I}{\partial x \partial t} + \frac{\partial^2 I}{\partial x^2} \right) \tag{3.21}$$

Uras *et al.* [3] calculate the partial derivatives of (3.19) from the image sequence and solve for u and v by finding the inverse of H. The dependence of second order techniques on the numerical second derivatives leaves them particularly susceptible to noise. Although smoothing the image intensities can reduce the effects of noise, techniques for identifying and discarding erroneous values are usually required.

This technique is also a well known and popular motion estimation technique [35, 106, 136], and is the third example technique used in this thesis to test the Information Fusion approaches and is refered to as URAS.

3.3.4 Hierarchical Approaches

Motion can occur at many scales in a image sequence and so it makes sense to estimate the general motion in scenes at coarser scales and refine these estimates with information from finer scales. Such *hierarchical* approaches have been applied both to differential and to correlation based Motion Estimation techniques. Glazer *et al.* [141] describe a hierarchical version of Horn and Schunck's [2] differential technique which uses a Gaussian low pass pyramid to construct views of the scene at different resolutions.

Anandan [142] proposes a framework for hierarchical motion estimation which consists of three stages. In the first stage, images are filtered into low and high frequency resolutions. In the second stage, which Anandan terms the *matching step*, motion is calculated calculated at the coarsest scale using some preferred Motion Estimation strategy, given the application at hand. In the final stage, the Motion Estimates from coarser scales are projected onto finer scales then refined. The second two stages are iterated until motion estimates are known for the finest scale. Anandan uses a correlation based approach in the matching step, but notes that differential techniques such as that of Glazer *et al.* [141] are also compatible with the framework.

3.4 Example Techniques

This section presents experimental results from applying the three example techniques which are used later in this thesis to test information fusion strategies. The CAMUS technique is chosen as illustrative of correlation approaches to Motion Estimation while the HORN and URAS techniques are selected as illustrative of differential approaches. These techniques have been chosen over alternatives because they provide simple and intuitive implementations of the correlation and differential approaches, and are therefore a good illustration of the characteristics of the approaches.

3.4.1 Confidence Measures

The performance of a Motion Estimation algorithm can vary considerably depending on the image sequence being processed. This performance can vary even over regions of the image sequence. For example, correlation based techniques perform poorly where there are few features to match to, so in the moving photo sequence results are very poor on the background but good on the moving photo.

Motion estimates are known to be error prone and much attention has been paid to providing confidence measures [143, 144]. Differential approaches require numerical evaluation of the image intensity derivatives, and typically confidence measures for differential techniques are based on these derivatives [107]. A comparison of confidence measures by Bainbridge-Smith *et al.* [143] suggests that the magnitude of the determinant of the Hessian of the spatial intensities is a good confidence measure, with larger values suggesting better accuracy of the motion estimates.

Confidence measures have been proposed for block matching techniques which are based on the topology of the similarity measure values [36]. An example distortion surface topology is illustrated in Figure 3.3. If the surface is flat, then this would suggest that there is little to distinguish between possible best match positions and the motion estimate is likely to be poor. A simple measure of the flatness of the topology is the variance of the similarity values.

Confidence measures are often used by Information Fusion strategies to gauge the reliability of individual results and so confidence measures are also described for the three example techniques presented in this section.

3.4.2 Accuracy-Efficiency Trade-Off

As reported by Liu *et al.* in [136], there is often a trade-off between the accuracy and efficiency of motion estimation techniques. The trade-off arises because more accurate techniques tend to have greater computational requirements. Many techniques have parameters which can be varied to influence the trade-off, for example the time range considered in CAMUS or the maximum number of iterations in HORN can be increased to improve accuracy but at the cost of greater time requirements [52]. Liu *et al.* characterise this trade-off by the *Accuracy-Efficiency* (AE) curve, which plots an error measure on the X-axis against a run-time measure on the Y-axis. Given similar computational resources, those techniques to the top left of the AE-space will make less frequent but more accurate estimates of motion than those to the bottom left.

In order to calculate the AE curve for a given Motion Estimation algorithm, the algorithm must be run on a data sequence for which the ground truth motion is known. Synthetic data sequences, such as the Diverging Tree sequence which was used by Liu *et al.* in their study, can be used for this purpose. Figure 3.10(a) shows an example frame from the Diverging Tree sequence with the true motion for the frame illustrated in Figure 3.10(b). It is harder to determine the ground truth for real data sequences. Figure 3.10(c) shows a sample frame from a simple experimental test sequence where a photograph moves from right to left across the frame at a constant speed. Ground truth motion for this simple Translating Photo sequence can easily be calculated manually, and is shown in Figure 3.10(d). A description of this sequence is given in Appendix C.

A popular measure of accuracy is the difference measure used in the survey of Motion Estima-

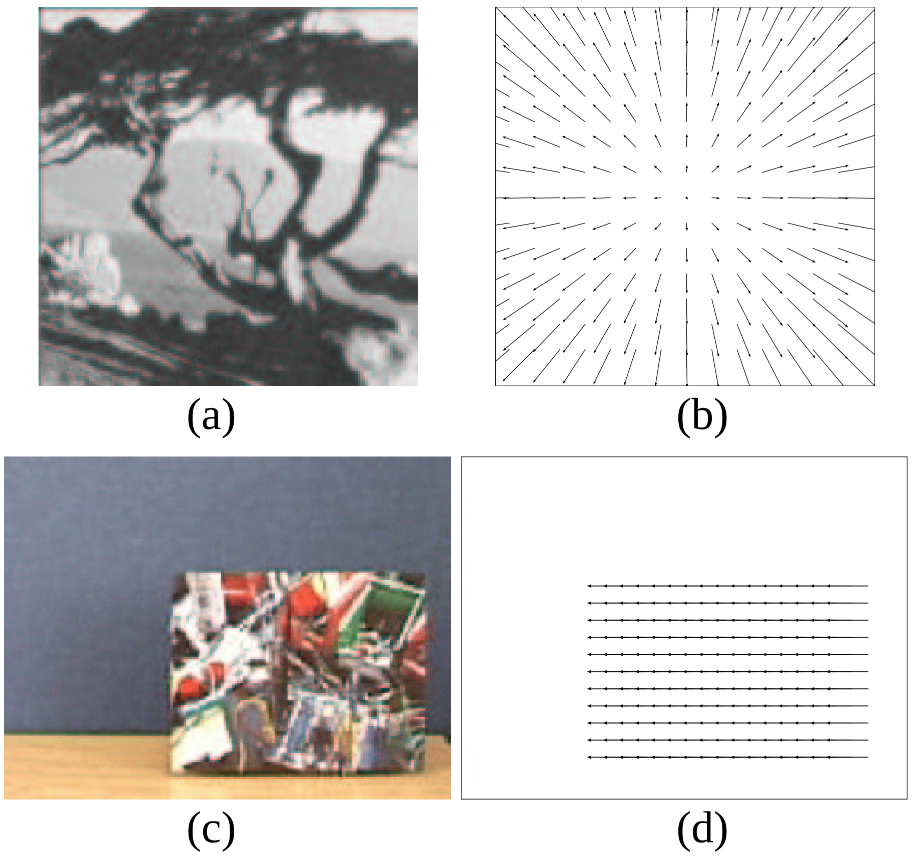


Figure 3.10: *(a) Image from diverging tree sequence, (b) correct motion from frame. (c) Image from translating photo sequence, (d) correct motion for frame.*

tion techniques by Barron *et al.* [107]. This measure was also used by Liu *et al.* [136] in their work on the Accuracy-Efficiency trade-off, and so is used in this thesis. This measure interprets 2D motion vectors as 3D vectors where the third dimension is time, set as 1 to indicate motion per frame. The difference measure is the angular error between the correct motion vector $\mathbf{u_c} = (u_c, v_c, 1)^T$ and the estimated vector $\mathbf{u_e} = (u_e, v_e, 1)^T$:

$$\text{angular error} = \arccos(\mathbf{u_c} \cdot \mathbf{u_e}) \tag{3.22}$$

It is difficult to estimate algorithm 'efficiency' empirically, as run time can be very dependent on the system architecture. For the AE curves described in this thesis, a theoretical estimate of complexity was given using O notation. $O(g(n))$ describes the asymptotic upper bound of a function and is defined as follows [145]:

$$\begin{aligned} O(g(n)) = \{f(n) \quad : \quad & \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \\ & \text{for all } n \geq n_0\} \end{aligned} \tag{3.23}$$

where $g(n)$ and $f(n)$ are functions expressing the computational requirements, in this case run time, of an algorithm with respect to a variable n, for example block width in the BMA.

The O complexities were also verified empirically by running the algorithms on a SUN Ultra60 with 768M RAM and twin 295MHz Sun UltraSPARC-II processors. All the implementations discussed in this thesis were coded by the author in C++. Appendix D describes the coding style and conventions employed. The remainder of this section presents and discusses AE curves calculated for the CAMUS, HORN, and URAS techniques.

3.4.3 CAMUS

The CAMUS technique is a correlation approach to Motion Estimation using an extension of the Block Matching Algorithm with a search over t subsequent frames. The accuracy of the CAMUS technique is mainly affected by the block size, which determines what features can be matched, and the time range, which enables sub-pel accuracy in the motion vectors. The search

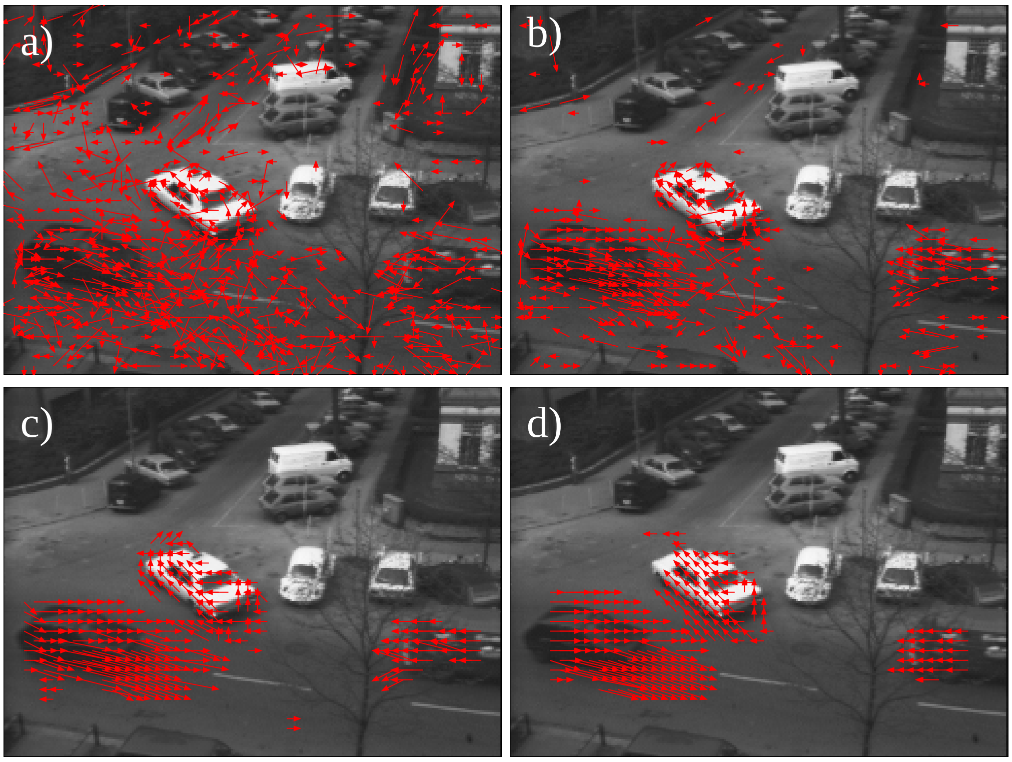


Figure 3.11: *Results (at frame 10) of applying the FSA with MAD distortion function to the Hamburg taxi sequence. Block size a) 4 $\times$ 4 b) 8 $\times$ 8, c) 16 $\times$ 16, d) 32 $\times$ 32.*

radius (w) should be fixed to the maximum motion expected in the sequence.

3.4.3.1 Varying the Block Size

Figure 3.11 illustrates the effects of using different block sizes when applying the FSA with AD distortion function to the Hamburg Taxi sequence. In each case the search area around the block was fixed at $w = 5$ pels. Motion vectors have been calculated for each pel by applying the CAMUS technique to a block centred on the pel. It can be seen from these results that where the block size is too small, the motion vectors tend to be very inaccurate, when they are too large, pels near moving objects are erroneously reported as being part of the moving object.

Figure 3.12 shows AE curves derived from applying the CAMUS technique to the Diverging Tree sequence and the Translating Photo sequence while varying the block size with the time range and spatial search radius fixed. Results are shown for the time range fixed at both 1 frame and 5 frames. In all cases the spatial search range was fixed at 5 pels.

From (3.1) it follows that the cost of estimating the motion of a square pel block of width N for a search radius w is:

$$T_D = (2w+1)^2 \times (N^2 \times T_f) = O(N^2) \tag{3.24}$$

where T_f is the time required to compute f. This was used to calculate the theoretical complexity estimates in Figure 3.12. A number of observations can be made:

- Larger block sizes tend to give smaller errors but have larger computational complexity and so results for large block sizes are towards the top left of the AE graph. Results for smaller block sizes lie towards the bottom right of the graph.
- AE trade-off performance on the artificial data sequence is better than on the real data sequence, in that the AE curves come closer to the origin. Curves which come close to the origin offer both low error and high speed.
- The smallest error rates are obtained on the real data sequence. This is partly because the motion in the diverging tree sequence is not constant over any blocks, and so the assumptions underlying the CAMUS algorithm are not valid.
- Although the theoretical and empirical graph shapes (green and red lines respectively) are similar, the theoretical complexity tends to underestimate the empirical run time requirements, especially for small block sizes. This is due to the less significant terms in (3.24).

Performance on the real data sequence was found to be particularly poor on the background where there are few features for the similarity measures to work with.

3.4.3.2 Varying the Time Range

Figure 3.13(a) shows the AE curve from applying the CAMUS technique to the diverging tree sequence varying the time range from 1 to 5 frames. For this experiment, the spatial search was again set at 5 pels and the block size 5×5 pels. The theoretical complexity is linear in the time range, or $O(t)$. A number of observations can be made:

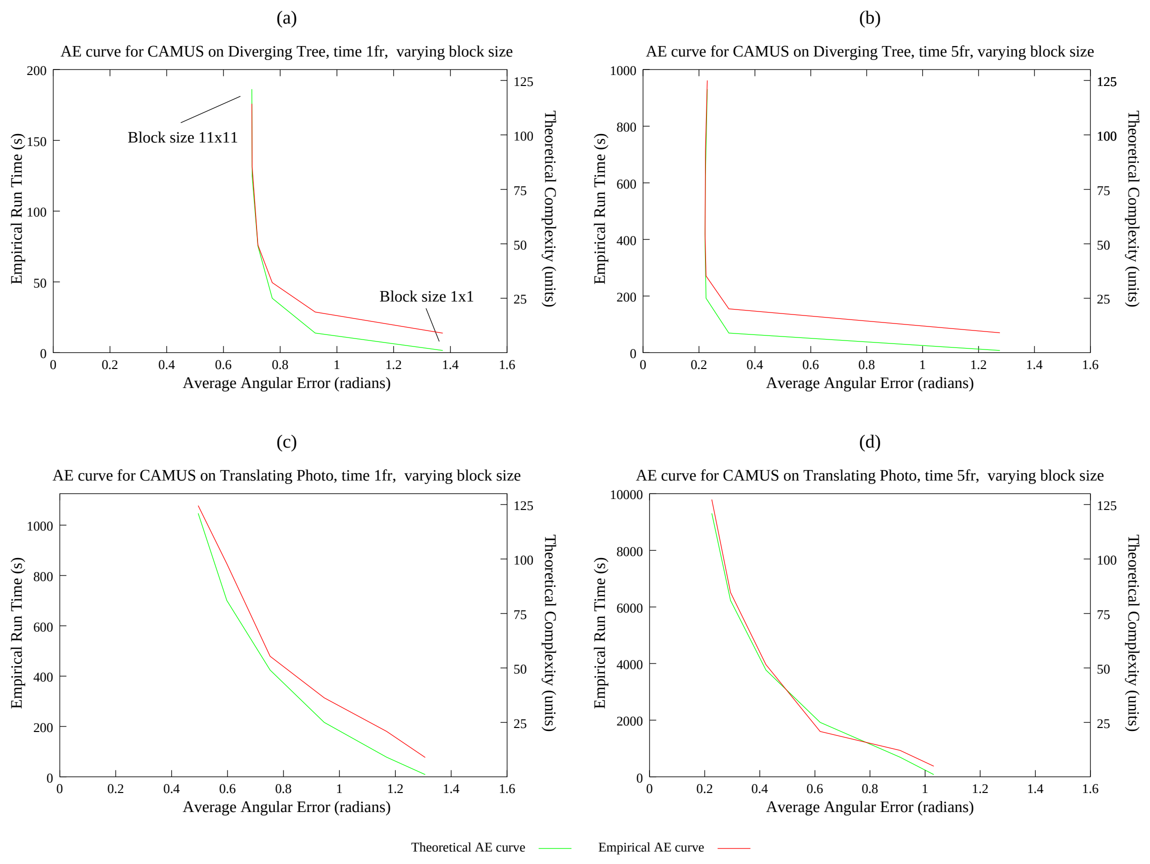


Figure 3.12: *AE curves from applying the CAMUS technique to a) the diverging tree sequence with time range 1 frame, b) the diverging tree sequence with time range 5 frames, c) the translating photo sequence with time range 1 frame, d) the translating photo sequence with time range 5 frames, in all cases varying the block size from* 1×1 *to* 11×11. *Curves calculated using theoretical computational requirements are shown in green, curves using empirical computational requirements are shown in red.*

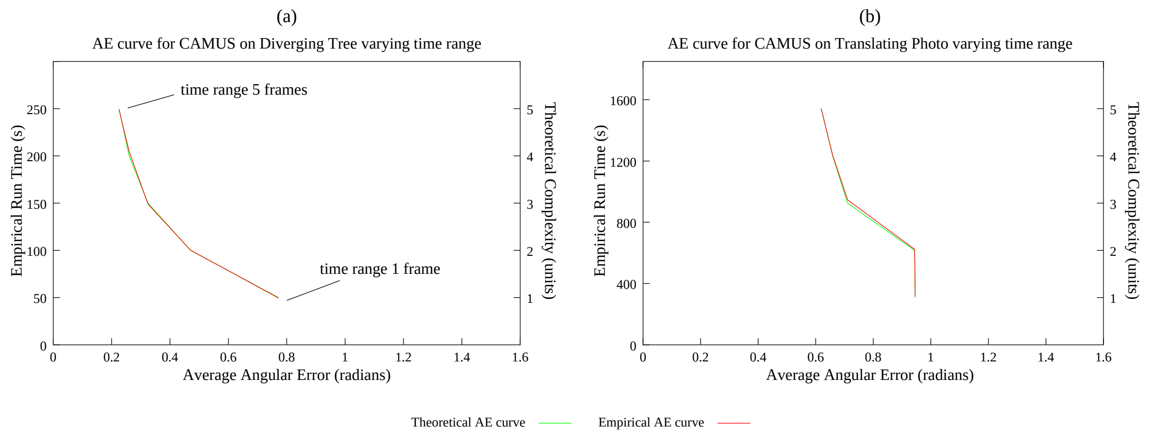


Figure 3.13: *AE curves from applying the CAMUS technique to a) the diverging tree sequence and b) the translating photo sequence varying the time range from* 1 *to* 5 *frames.*

- Larger time ranges tend to give smaller errors but have larger computational complexity. With a single frame time range the precision is limited to the nearest pel, as this is increased the available precision increases and the accuracy increases accordingly. If the time range is increased too far, however, a decrease in performance may occur due to more incorrect matches being available.

- The theoretical and empirical curves are very similar, this shows that the theoretical run time is a good estimate of the true run time on this architecture.

- Accuracy is greater on the synthetic data than on the real data, with the most accurate results for the synthetic and real data being approximately 0.2 radians and 0.6 radians respectively. The results in Figure 3.12 showed that with 11×11 block sizes CAMUS is more accurate on the real data which better matches the assumptions underlying the technique. However, the effects of image noise in the real sequence are greater on 5×5 pel blocks than on 11×11 pel blocks, which leads to this change in relative performance.

- The angular errors on the Translating Photo sequence are very similar for 1 and 2 time steps, which leads to an unusual shape of the curve in Figure 3.13(b). This is because a single time step does not give enough time for the small motion vectors to become apparent, and the majority of motion estimates are simply “no motion”.

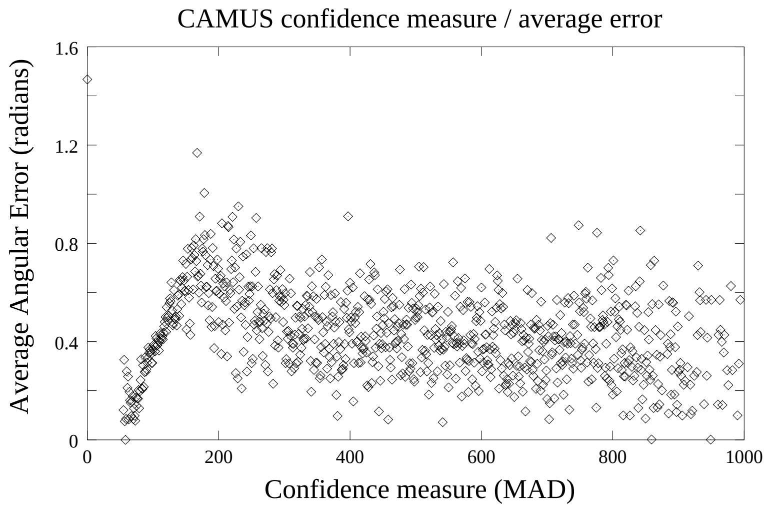


Figure 3.14: *Plot of a confidence measure for CAMUS based on the variance of the AD similarity measure against the average angular error.*

3.4.3.3 Confidence Measures

Local statistics of the value of the similarity measure used for block matching can be used as a confidence measure for CAMUS [36]. A low variance in the measure might suggest that there is little difference between a number of possible motion vectors, and so the motion estimate is likely to be inaccurate. Figure 3.14 shows the variance of the AD similarity for a matched block against the average error using a block size of 7×7 pels, a search radius of 3 pels and a time range of 5 frames on the Translating Photo sequence. It can be seen that there is an almost linear relationship at first but larger values are less meaningful.

3.4.4 HORN

The HORN technique is a first order differential approach to Motion Estimation. Results from applying the HORN method to the Hamburg Taxi sequence with $\lambda = 100$ are shown in Figure 3.15. Figure 3.15(a) shows the result after 10 iteration of the iterative solution and Figure 3.15(b) shows the result after 200 iterations, the effects of the smoothness constraint can be seen in the differences between these results. Over regularisation due to the smoothness constraint can lead to regions of no motion between moving regions to be reported with small motion vectors. Thresholding is often used to remove small vectors assumed to be due to this over-regularisation.

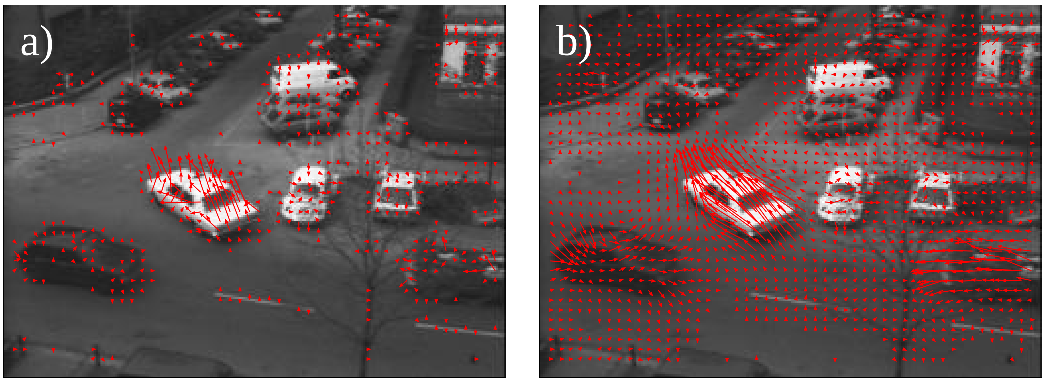


Figure 3.15: *Results from applying the HORN Motion Estimation technique to the Hamburg Taxi sequence, a) results after 10 iterations and b) results after 200 iterations.*

Liu *et al.* report the HORN method as a point in AE space, however it can be seen that the trade-off between accuracy and speed, given the choice of the maximum number of iterations allowed, is an AE trade-off and so the technique has an AE curve. The theoretical run time is linear in the number of iterations I, or $O(I)$. Figure 3.16 shows the HORN AE curves derived using the diverging tree and translating photo sequence. To help reduce the effects of over-regularisation, motion vectors with magnitudes less than 0.1pel/frame were thresholded to no motion. The motion vectors were initialised to zero at each frame before running the algorithm.

λ is a weighting factor with larger values leading to a smoother motion field. Horn and Schunck suggest that λ should be proportional to the noise in the image sequence, but other than this the choice of λ tends to be empirical. A number of authors have suggested very different choices of λ, ranging from 0.5 to 100 [2, 107]. Figure 3.16 shows the AE curve results for $\lambda = 0.5$ (a,c) and for $\lambda = 100$ (b,d). A number of observations can be made:

- The theoretical and empirical curves are very similar, this shows that the theoretical run time is a good estimate of the true run time on this architecture.
- After 200 iterations, $\lambda = 100$ led to smaller errors than $\lambda = 0.5$. This is because the smoothness constraint is a reasonable assumption in both the image sequences. This result agrees with Barron *et al.* [107] who also ran the algorithm on the Diverging Tree sequence.
- For the first few iterations on the Diverging Tree sequence, $\lambda = 100$ led to larger errors than $\lambda = 0.5$, this was not noted by Barron *et al.* [107]. The initial poor performance is

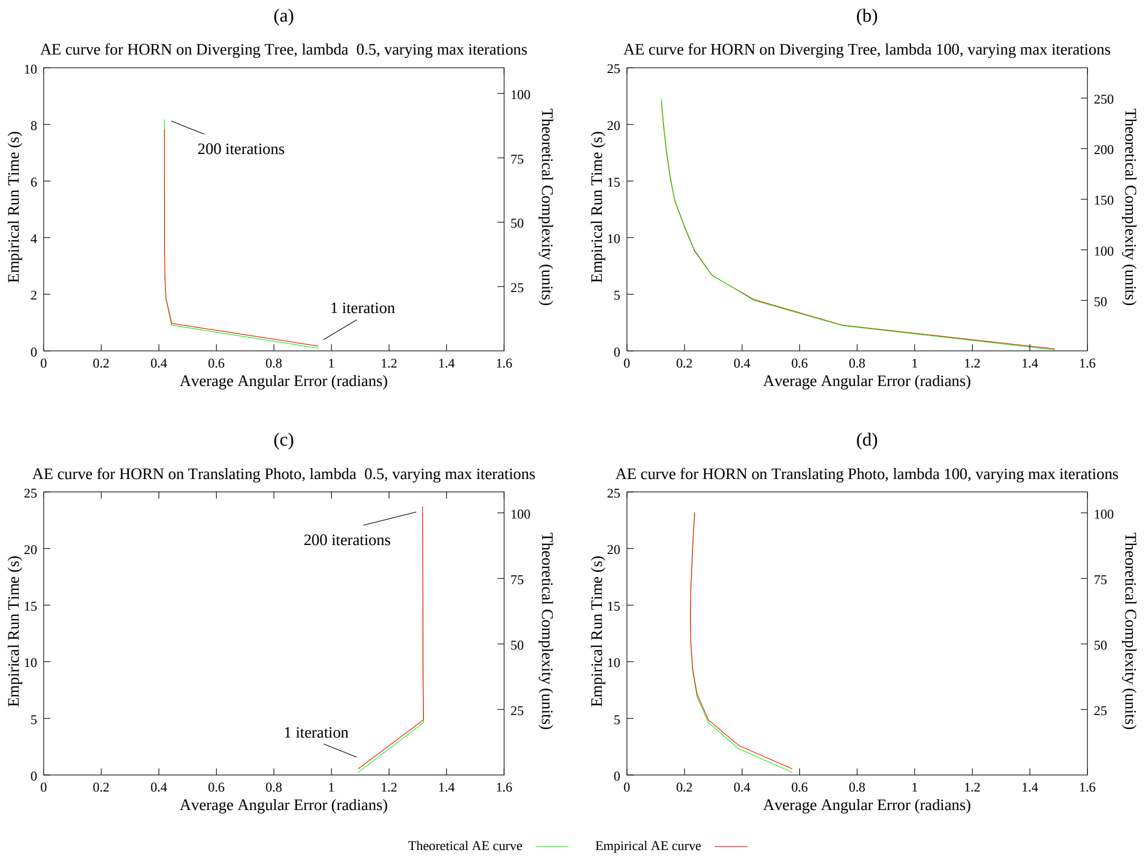


Figure 3.16: *AE curves from applying the HORN Motion Estimation technique to a) the diverging tree sequence with $\lambda = 0.5$, b) the diverging tree sequence with $\lambda = 100$, c) the translating photo sequence with $\lambda = 0.5$, d) the translating photo sequence with $\lambda = 100$, in all cases varying the number of iterations.*

due to the iterative nature of the algorithm. As described in Appendix B.1, the average horizontal $\bar{u}_{kl}$ and vertical $\bar{v}_{kl}$ components in Equation (3.18) are approximated by the immediate neighbours to the pel location. Therefore it takes a number of iterations for the smoothness constraint to have its intended effect.

- With $\lambda = 0.5$, HORN performs very poorly on the translating photo sequence. The relatively low error with 1 iteration is due to the initialised state of no motion being correct over a large portion of the image. This is due to noise in the images making it hard to accurately calculate the intensity derivatives.

Setting $\lambda = 100$ led to more accurate results on both test sequences. Barron *et al.* report that $\lambda = 0.5$ leads to better results, however they only allow the algorithm to run for 100 iterations. λ is a weighting factor on the smoothness constraint, and so for real image sequences where noise can be a problem it makes sense to use a larger value of λ. For these reasons, $\lambda = 100$ has been adopted for the experiments described in this thesis.

3.4.4.1 Confidence Measures

The residual error from (3.16) gives an intuitive confidence measure for HORN motion estimates. Figure 3.17 shows a plot of the residual error against the mean angular error for the translating photo sequence. It can be seen that although there is a trend towards large residual errors suggesting larger angular errors, there are many outliers.

3.4.5 URAS

The URAS technique is a second order differential approach to Motion Estimation. Figure 3.19(d) shows a typical result from the Hamburg Taxi sequence using the URAS technique. The images were smoothed using a Gaussian kernel of standard deviation 5 pels spatially and 1 pel temporally. The second derivatives were found using cascaded 4 point intensity differences.

Uras *et al.* use the condition number of the Hessian $\mathbf{H}$ as a confidence measure. Barron *et al.* [107] suggest that the determinant of $\mathbf{H}$ is also a good confidence measure. Bainbridge-Smith *et al.* [143] compare the determinant of $\mathbf{H}$ favourably to the condition number, and since this must be calculated anyway, this is the measure used in this thesis. Figure 3.18 shows a plot of the determinant of $\mathbf{H}$ against the mean angular error, illustrating that where the determinant

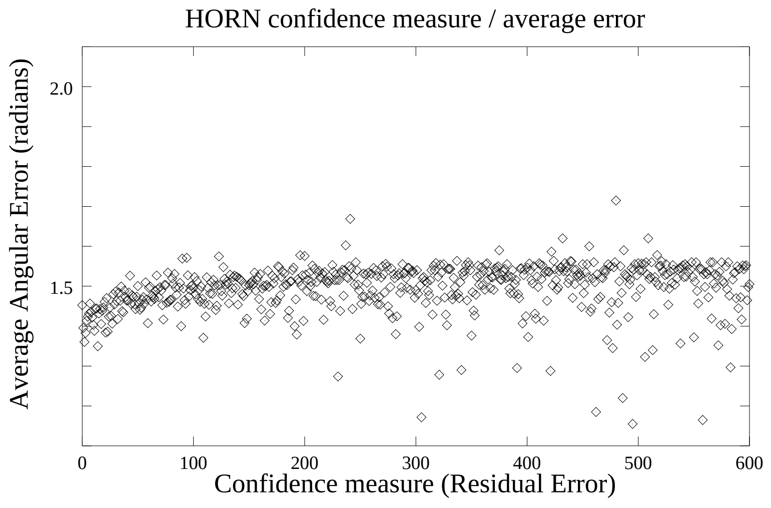


Figure 3.17: *Plot of a confidence measure for HORN against the average angular error.*

is small the error tends to be large. To reduce the occurrence of spurious motion vectors, the image is divided into blocks and for each block a representative vector is selected as the vector with the most confidence (largest $\det(\mathbf{H})$). A typical result using 9×9 blocks is given in Figure 3.19(c).

Vectors with very low confidence measures can be discarded as erroneous. Barron *et al.* suggest discarding any vector with $\det(\mathbf{H}) < 1.0$. A typical result from this is shown in Figure 3.19(a).

The application of temporal smoothing is particularly interesting as it means that the motion for a given frame cannot be determined until information from subsequent frames is made available. The need for temporal smoothing is illustrated by Figure 3.19(b), which shows the results without temporal smoothing.

3.5 Summary

This chapter has presented a review of image processing techniques for analysing motion in image sequences. Techniques have been classified according to whether they are being used for Motion Compensation in image compression, or for Motion Estimation where the aim is to determine the real world causes of image changes. This is a useful taxonomy because the paradigms underlying these classes are very different and so even when two techniques from different classes are based on similar concepts their implementations can differ significantly.

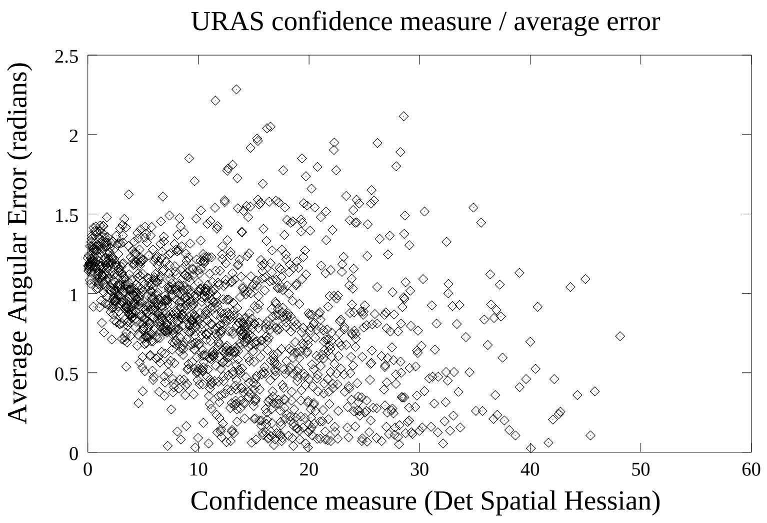


Figure 3.18: *Plot of a confidence measure for URAS against the average angular error.*

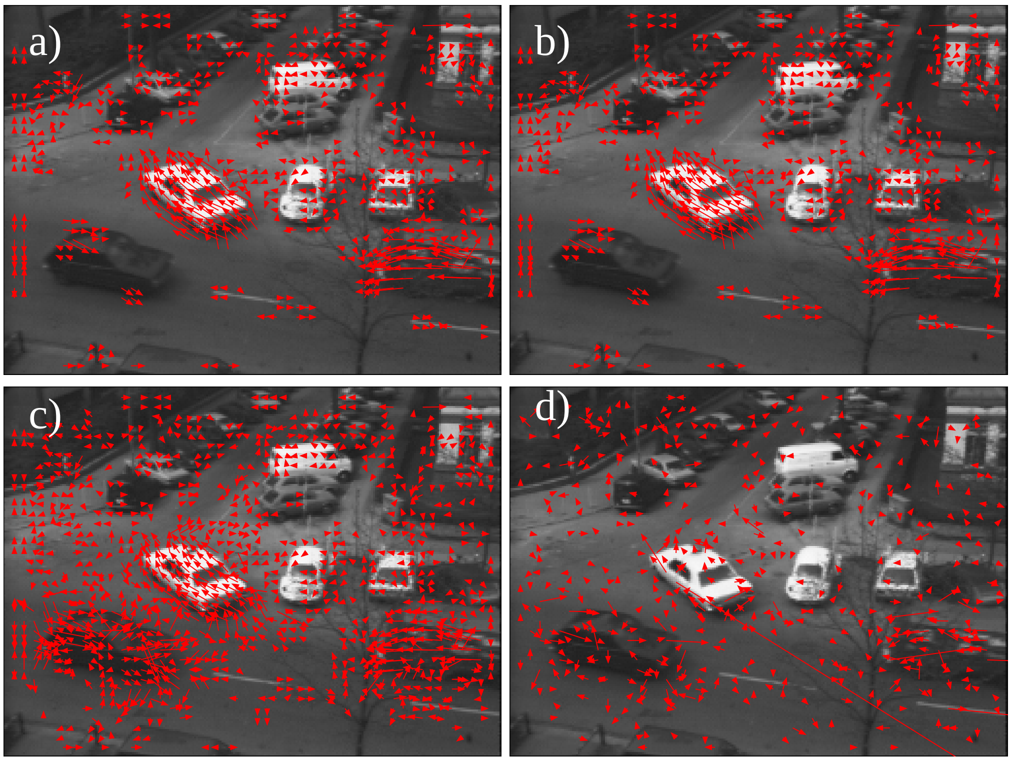


Figure 3.19: *Results (at frame 15) of applying URAS to the Hamburg taxi sequence. a) Typical result b) no temporal smoothing, c) no thresholding, d) No thresholding or best block representative.*

The theme of this thesis is the application of Information Fusion strategies to Motion Estimation. The two main groups of Motion Estimation techniques are the differential and the correlation based approaches. A few approaches also work in the frequency domain, this is more popular for Motion Compensation problems because a number of compression algorithms use transforms such as the DCT. This chapter also described a novel technique for fast though rough motion analysis in the DCT domain.

As the main classes of Motion Estimation technique are the differential and correlation based approaches, techniques from these two classes will be used in subsequent chapters to illustrate the effects of Information Fusion methods. Three popular Motion Estimation techniques, CAMUS, HORN and URAS, have been chosen for this purpose. CAMUS is a block matching technique that is illustrative of correlation approaches while HORN and URAS are differential methods.

The three example Motion Estimation strategies have been illustrated by applying them to synthetic and real test image sequences. The performances of the HORN and CAMUS strategies have been analysed in terms of the AE trade-off, with the computational requirements considered both theoretically and empirically. Previously HORN has only been reported in the literature as a point in AE space.

Methods for providing confidence measures for the Motion Estimation results have been identified and described. These will be necessary in subsequent chapters to enable the contributions of individual estimates to the fused estimate to be weighted appropriately.

The review presented in this chapter has shown that there are a wide variety of techniques for analysing motion in image sequences, and in particular for Motion Estimation. The performance of these techniques both in terms of accuracy and speed can be data dependent. Motion Estimation has a variety of applications, and so the provision of accurate, robust and timely estimates is an important problem. The remainder of this thesis will consider how Information Fusion might be used for this purpose.

Chapter 4
Fusion of Electrorotation Frequency Estimates

4.1 Introduction

The previous chapters have given an overview of the Information Fusion and Motion Estimation research fields. This chapter gives an example of the application of some of these ideas, by applying Information Fusion strategies to the problem of fusing estimates of the speed of rotation of rotating cells. This is a simpler fusion problem than will be considered in subsequent chapters, as the motion estimates are scalar values.

When a nonuniform electric field is applied to neutral biological cells, the polarisation effects can result in translational motion of the cells. This effect is called *dielectrophoresis* and is described by Pohl [146]. Careful arrangement of electrodes and their polarities can also be used to cause the cells to rotate, this effect is called *electrorotation*. The electrorotation of cells provides sensitive measurements of their physiological states [147, 148]. As a result, measuring the speed of rotation is an important problem with a a variety of commercial applications [149, 150]. For example, this can be used to determine cell viability and has been applied to evaluating water quality by measuring the states of cells exposed to the water.

Cells undergoing electrorotation can be imaged using a camera attached to a microscope. An example image taken from a typical electrorotation sequence is shown in Figure 4.1, the dimensions of the frame are 352×288 pels.

Although each cell is constrained to rotate about an axis that is normal to the viewing plane, there are inevitable perturbations. This, together with the fact that the cells are three dimensional and the limited depth of field of the imaging optics, produces image sequences that are not simple rotations of a two dimensional projection. Perturbations outside this 2D rotation are not of interest to the electrorotation analysis.

The frequency of rotation is currently estimated manually, which is both tedious and time consuming. Zhou *et al.* [147] report a method for automatically determining the speed of rotation

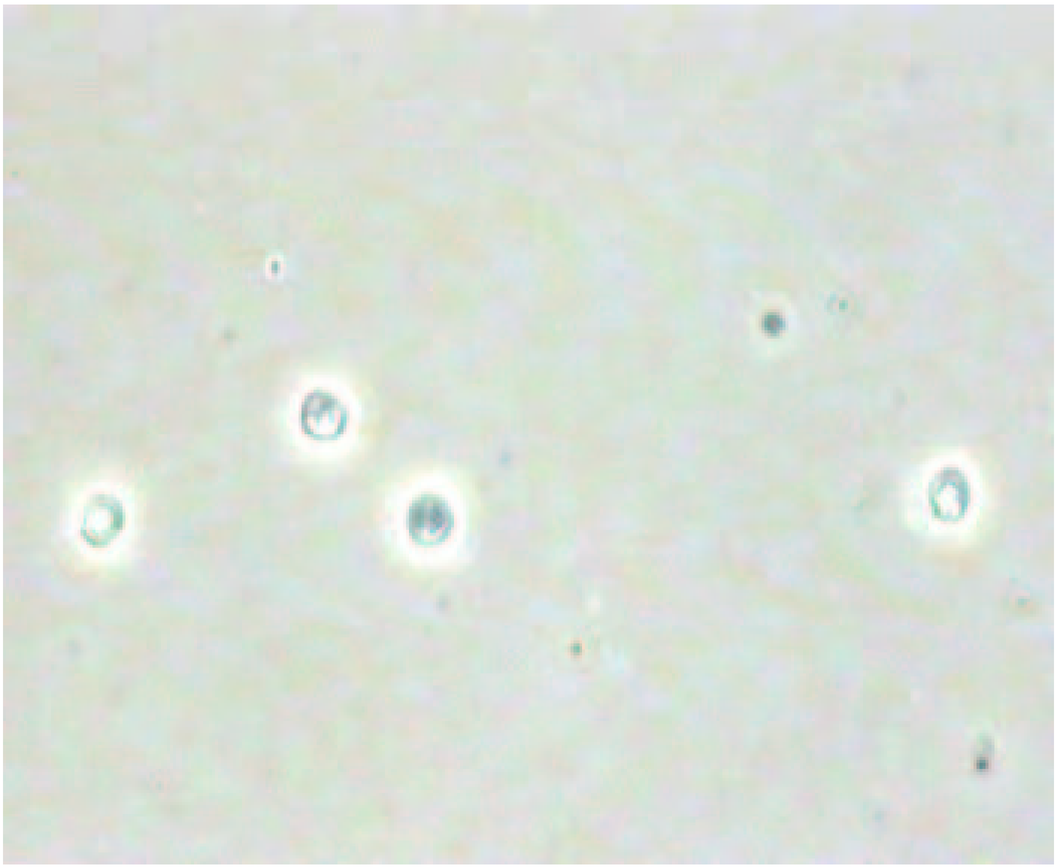


Figure 4.1: *An example frame from a typical electrorotation sequence.*

from images of the rotating cells. Their method relies on a good initial segmentation of cells from the images, which is used to find the principal axis of the cell. By examining the changing angle between the principal axis and the horizontal axis over time, the frequency of rotation can be determined. This method assumes that cells are stationary and non-circular; however, in practice cells usually also undergo some translation.

This chapter describes a novel method for automatically determining the speed of rotation from images of the rotating cells. The method consists of two parts, tracking the cells over a sequence of frames and then determining the frequency of rotation from the cell appearance over the tracked sequence. Three methods for determining the frequency of rotation are presented; the first is an extension of the principal axis based method of Zhou *et al.*, the second technique rotates tracked cell images and uses correlation techniques to determine the best angle of rotation. The final technique determines the frequency of rotation from the changing appearance of the cell with respect to a reference frame.

Although the individual techniques perform well on certain examples, none of the techniques perform adequately in all cases. As discussed in Chapter 2, Information Fusion strategies can be employed to combine the relative strengths of individual sensors and classifiers to provide systems which perform robustly over a more general range than the individual systems alone. In this chapter, estimation fusion is used to combine the results of the different individual electrorotation frequency techniques to produce a system which performs more robustly over the data set.

The remainder of this chapter is organised as follows: Section 4.2 presents the tracking method used to account for the translatory motion of cells. Section 4.3 describes the individual techniques for estimating the frequency of rotation of the cells, results for the individual techniques are presented in Section 4.4, and Section 4.5 describes the fusion process and presents results from combining the individual estimates. Section 4.6 gives a summary of the chapter with some conclusions. The work presented and discussed in this chapter is reported in [54] and [49].

4.2 Tracking the cells

Frames from a typical electrorotation image sequence are shown in Figure 4.2. As can be seen from this figure, there are often multiple similar cells in a given image sequence. Clearly the cells can translate as well as rotate, and so a tracking system is required to remove the translat-

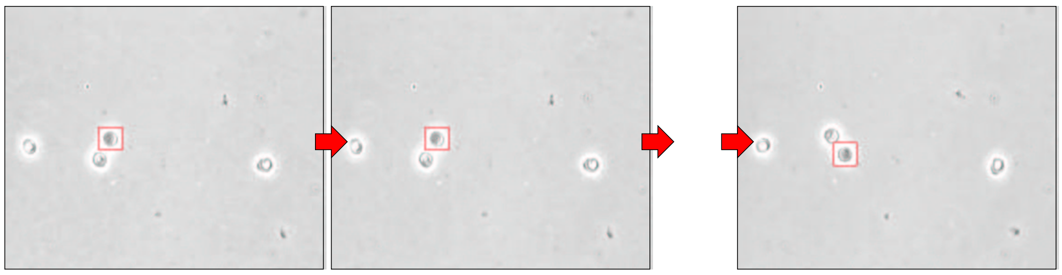


Figure 4.2: *Typical electrorotation frame sequence showing the first, second and last frames with tracked block.*

ory motion. The cells move independently and so can be tracked and analysed individually.

The Block Matching Algorithm (BMA), described in Chapter 3, is a popular correlation-based approach to tracking objects [37], and was adopted for this problem. In this approach, it is assumed that the appearance of a block of pixels will vary slowly over time. An object can, therefore, be tracked from frame to frame by taking a reference block of pixels on the object image in the current frame and searching for the most similar block of pixels in the next frame. A popular similarity measure between a square $(N \times N)$ reference block R and a pixel block B in the next frame is the Absolute Difference (AD):

$$\text{AD} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} |I_R(x,y) - I_B(x,y)| \tag{4.1}$$

where $I_R(x,y)$ and $I_B(x,y)$ are the intensities of the pixels at index (x,y) of blocks R and B respectively. The new location of the tracked reference block in the next frame is that of the pixel block B which yields the smallest AD value. As the speed of a tracked object is typically limited, the search can be restricted to a window around the original reference block location.

In the BMA [116], it is assumed that the appearance of a block of pels remains constant over time and motion. This is a reasonable assumption given high frame rates and short time periods. The motion of a block of pels is estimated by looking for the most similar block in subsequent frames according to some similarity measure. When the BMA is used for tracking, the reference block must be altered to take account of changes in the appearance of the target object. This extended algorithm is termed the Adaptive Block Matching Algorithm (ABMA).

There are a number of choices in the ABMA including the block size, the block shape, the similarity measure, selection of initial block and the reference block update strategy. The search is typically constrained to a window whose size must also be chosen. Errors in tracking can be due to numerous causes including camera noise and inappropriate choice of these parameters. However, as the electrorotation image sequences are captured under strict laboratory conditions, background noise is small and has little effect on the tracking and rotation analysis. The average background noise estimated over a typical image sequence with 256 grey levels was found to have a variance of just 0.8 grey levels.

Occlusion can also be a problem for tracking algorithms, as once the target is hidden from view the track is quickly lost. However, in the electrorotation sequences the cells are constrained to lie on a 2D plane and so occlusion was not found to be an issue.

Sub-pel motion can also cause the track to be lost: the problem is due to small changes in object appearance being insufficient to cause the track position to follow the motion, meanwhile the reference block is updated and so slowly comes to no longer resemble the target. The reference block updating strategy is an important problem, both in the electrorotation example and in general. This has not received much attention in the past and so is considered in detail next.

4.2.1 Test Data Sets

For this investigation of reference block updating, test images from more typical tracking problems were chosen. Four test data sets are used in the following section to enable a comparison of different update strategies. These are the well known Hamburg Taxi sequence, two sequences from a car park security camera, and a sequence taken from one car following another. The Hamburg Taxi sequence is too short to be informative, but is included to assist the reader in replicating the results. The car following and car park sequences are new sequences which are described in Appendix C. Since only the reference block update strategy is considered here, the sequences were chosen to have none of the missing or unintelligible frames which can appear in real data sequences.

In the car following data set, both the target and the camera are in motion and so there are a lot of small quick movements due to uneven roads. Car park sequence A shows a pedestrian walking, while B shows a car moving behind a fence. Coming from a real security camera rather than controlled laboratory conditions, the car park data set is quite poor quality and so no single reference block is ever a particularly good representation of the target. Due to format conversions when capturing the data, there are also some blocking artifacts present in the sequence, these are described in Appendix C. Figure 4.3 shows typical frames from the data sets, with arrows indicating the track points. In these three data sets, the appearance of the target varies considerably over the sequence.

For each data sequence, a point on the target object was chosen and its position, at 5 frame intervals, was tracked manually over the sequence. The manual estimates are accurate to approximately 1 pel. These measurements are used as ground truth to enable comparison of the strategies discussed below. The mean errors and error variances for all tested methods, with respect to this manual tracking, are given in Table 4.1.

4.2.2 Update Strategies

This section describes a number of update strategies and discusses results from applying these strategies to the test sequences described above. The results presented here are all from square blocks of width 8 pels. The AD similarity measure was used. Appropriate search windows and start blocks were manually chosen for each sequence.

Single Frame Strategy The simplest update strategy is to update the reference block every T_S frames by replacing it with the block at the current track position. With $T_S = 1$, the reference block is replaced every frame, with $T_S = \infty$, the block is never replaced. If the reference block is changed too frequently then the small errors due to camera noise and rounding error accumulate and can cause the tracking to be lost. If it is not changed frequently enough, then the appearance of the object may change too much for a good match to be found and again the tracking may be lost.

As the results in Table 4.1 show, the performance of this strategy is highly dependent on the choice of T_S. The large error and variance measures on the Car Park A sequence with $T_S = \infty$ are due to the track being lost during the sequence but regained when the object position and the track estimate happened to coincide later on. In all cases, setting $T_S = 1$ led to poor performance. Large values of T_S require long term memory, which is problematic for analogue implementation.

Although results for this are not presented here, larger block sizes contain more information, so reducing the effects of noise and enabling the reference block to be changed more frequently. However, the complexity of the algorithm is quadratic in the block size. Also, large block sizes can often include a lot of background image, which can be problematic if the reference block starts to match to the background rather than to the tracked object.

Multi Frame Strategy The Multi Frame strategy uses T_M reference blocks, which are the blocks at the track positions of the previous T_M time steps. For each search position, the similarity function f is applied separately with all reference blocks. The similarity measure f_{MULTI} for a given position is a weighted sum of these values. Since older matches are less likely to correspond to the current block, a weighting strategy which prefers more recent blocks is appropriate. The results presented here use $T_M = 5$ with weights $w_i = 0.9^{i-1}$:

$$f_{MULTI}(B_t, M_{t-1}, \ldots, M_{t-T_M}) = \sum_{i=1}^{T_M} w_i f(B_t, M_{t-i}) \tag{4.2}$$

where M_i is the best matched block at time i and B_t is block at time t being checked for similarity. This strategy requires that memory be provided for the T_M reference blocks, and also requires $T_M - 1$ additional matching operations. The results in Table 4.1 show that this approach performs better than the Single update strategy.

FIR Strategy The use of FIR filters for tracking purposes is well known [98]. Here the filter is used to track the changing block appearance rather than the object's coordinates. The filter is described below in its intuitive non-recursive form for F non-zero coefficients a_i:

$$R_t = a_{\text{NORM}} \sum_{i=1}^{F} a_i M_{k-i} \tag{4.3}$$

where R_t is the reference block at time t. So the reference block at any time is dependent on a linear combination of the last F blocks. $a_{\text{NORM}} = \sum_{i=1}^{F} a_i$ is a normalisation term used to prevent lightening or darkening of the reference block. Typically, $a_0 = 1$ and the coefficients get smaller over time so that older reference blocks count less than more recent blocks. Thus the filtered reference block adapts to the changing target appearance, but is stable in the face of noise. The results presented here use $F = 5$ and $a_i = 0.9^{i-1}$.

Again, memory must be provided for the F previous blocks. Only one matching operation is required, but an additional F block multiply-accumulate operations must be performed. The results in Table 4.1 show that in general this approach performs better than the Single and Multi Frame update strategies.

Kalman Filter Strategy Although the Multi Frame and FIR Frame strategies are more effective than the Single Frame strategy, they also incur greater computational cost. In both cases, the computational complexity of the additional operations is linear. The use of a Kalman filter [98] (see Appendix B.2) can reduce the additional cost to a small constant.

In this strategy, the $N \times N$ reference block at time t is represented by the N^2 state vector $\mathbf{r}_t$ and the observation at time t is the best match block represented by the N^2 vector $\mathbf{m}_t$. As the state and observation space are identical, the conversion from state space to observation space H_n is just the identity matrix. Since the BMA model assumes that the appearance of a block of pels is constant over time, the state transition matrix Φ is also the identity matrix. If it is assumed that the dynamic model noise and measurement error parameters are constant, then the Kalman Gain K can be calculated off-line [98] by running the filter until K becomes constant. Under these assumptions, only a single update equation for the reference block is required:

$$\mathbf{r}_t = \mathbf{r}_{t-1} + K(\mathbf{m}_{t-1} - \mathbf{r}_{t-1}) \tag{4.4}$$

thus the additional cost is very small. The results in Tables 4.1 show that this is the most accurate and robust of the tested strategies. Only a single reference block is maintained which is updated at each frame, and so the memory requirements are also relatively minor.

4.2.3 Tracking Cells

The Kalman filter update was found to be the most robust of the different reference block update strategies and so was used to track cells in the electrorotation test sequences. A typical track sequence from the electrorotation data set is illustrated in Figure 4.2, with the position of the

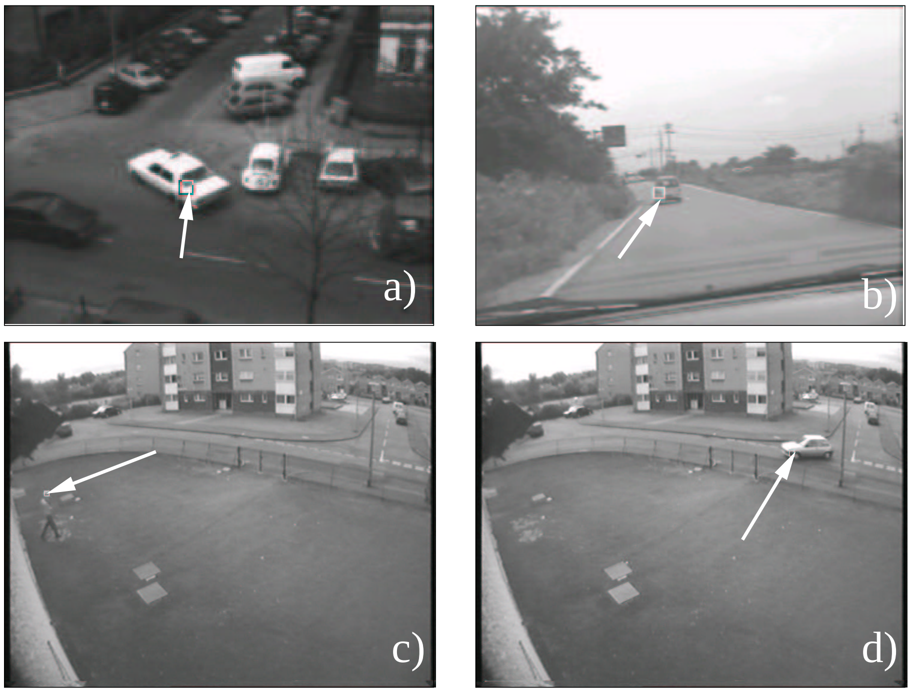


Figure 4.3: *Example frames from track sequences (not to same scale) with arrows indicating tracked points, (a) Hamburg Taxi, (b) Car Following, (c) Car Park A (pedestrian), (d) Car Park B (car)*

Sequence	Single $T_S = \infty$	Single $T_S = 5$	Single $T_S = 1$	Multi	FIR	Kalman
Hamburg Taxi	2.6 (0.5)	2.8 (0.5)	2.3 (0.3)	2.0 (0.2)	2.0 (0.2)	2.0 (0.2)
Car Following	LOST	3.4 (1.1)	LOST	LOST	3.7 (1.9)	2.8 (0.9)
Car Park A	10.5 (24.0)	2.0 (0.7)	LOST	2.3 (0.7)	2.3 (0.9)	1.7 (0.5)
Car Park B	4.2 (6.3)	4.1 (4.1)	LOST	2.9 (3.0)	2.5 (1.9)	1.2 (0.3)

Table 4.1: *Mean Tracking errors (in pels) with variance in parentheses, “LOST” indicates track lost.*

reference block outlined in red. In most tracking applications it is also necessary to filter the location estimates. However the performance of the BMA with a Kalman filter update strategy was found to be within 1 pel of the manual estimate on the electrorotation sequences and so it was not found necessary to use a filter on the cell location estimate.

In the final system, track initiation was performed by a human observer selecting a particular cell to analyse by giving its start position and appropriate block size to the tracking subsystem. Typical frame, reference block and search window dimensions are 352×288 pixels, 30×30 pixels and 40×40 pixels respectively. The tracking algorithm was applied to the electrorotation sequences to extract sequences of cell subimages with the translatory component of the motion removed. The techniques described in the following sections could then be applied to determine the frequency of rotation.

4.3 Counting Rotations

This section describes three techniques for determining the frequency of rotation. The first technique is that of Zhou *et al.* [147] based on extracting the Principal Axes of cells. The second technique artificially rotates the tracked cell images and uses correlation to determine the angle of rotation. The final technique determines the frequency of rotation from the changing similarity values with respect to a common reference block.

4.3.1 Principal Axis Method

Zhou *et al.* [147] segment cells from the background and find their principal axes, which are then used as a feature for determining the frequency of rotation.

In the data sequences examined by Zhou *et al.* the cell borders tended to be darker than the background and so could be segmented by thresholding. Any region totally surrounded by segmented pels was then filled, thus segmenting the cell centre. An erosion operator [130] was used to remove spurious pels. Finally, the (binary) segmented image was convolved with the original image to segment the cell, as illustrated in Figure 4.4. In the data sequences considered in this chapter, cell borders were frequently not darker than the background, so the SUSAN [151] edge detector was used rather than thresholding to detect the cell boundary in the first stage. The SUSAN operator, described in Appendix B.3 was chosen in preference to

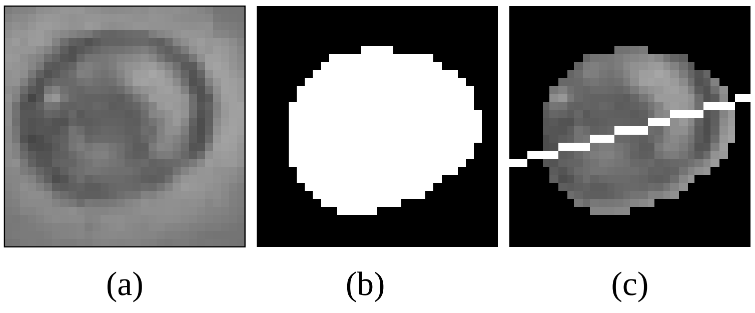


Figure 4.4: *(a) A tracked block, (b) an extracted cell position, (c) the segmented cell with Principal Axis.*

other edge detectors because its performance is invariant to the edge direction, which is useful for detecting the circular shaped cell edges.

The principal axis of a segmented cell is the most significant eigenvector of the 2×2 covariance matrix C of cell pel positions $\mathbf{x} = (x, y)$:

$$C = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y)(\mathbf{x}\mathbf{x}^{\mathrm{T}} - \mathbf{m}\mathbf{m}^{\mathrm{T}}) \tag{4.5}$$

where $\mathbf{m}$ is the mean position. By using the pel intensities $I(x, y)$to weight the contribution of the pel positions to C, the principal axis is made sensitive to features inside the cell. Figure 4.4(c) shows the principal axis of a segmented cell. As the cell rotates, so does the angle between the principal axis and the horizontal axis. The frequency of rotation is calculated from the average difference of this angle between successive frames.

4.3.2 Rotation Correlation Method

By rotating the image of a cell and comparing the result with the image of the cell in the next frame, an estimate of the angle of rotation can be determined. The degree of similarity between the artificially rotated cell and the next cell can be measured using the AD, however the AD is sensitive to intensity shifts and so the Normalised Correlation Coefficient (NCC) [130] was used (see also Equation 3.12). To create an image of the cell rotated by an angle θ, the original position (x', y') of each pixel (x, y) in the rotated block was found by multiplying by the Rotation Matrix [152]:

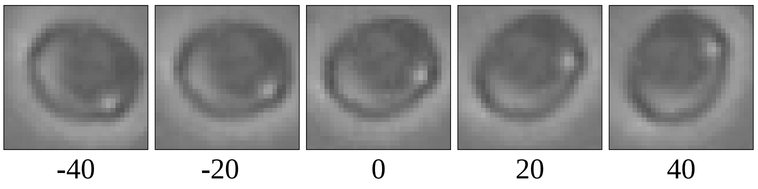


Figure 4.5: *Rotated cell images at angles of* $-40°, -20°, 0°, 20°$ *and* $40°$.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.6}$$

Since the images are discrete, (x', y') typically lies between pixel locations in the original image. The intensity of the rotated image pixel (x, y) was estimated by bilinear interpolation of the pixels around (x', y') in the original frame [152]. Example results of rotating a cell are shown in Figure 4.5.

For each frame, NCC similarity values for integer rotation angles in $1°$ steps from the range $[-90°, 90°]$ were extracted. These values were averaged over the entire image sequence and the angle for which the similarity was greatest was chosen.

4.3.3 Periodic Similarity Method

As a cell rotates, its appearance changes. Ideally, after a complete rotation the appearance will be the same. By looking at how the cell's appearance changes over time, the frequency of rotation may be estimated. Changes in appearance are determined relative to a reference block R. This reference block is simply chosen to be the first tracked block. The change in appearance between the current track block and the reference block is measured by the NCC.

Figure 4.6 shows the NCC values resulting from a typical electrorotation sequence. Peaks $t_n, n = 0, \ldots, N-1$ in the graph correspond to points where the cell completes a rotation. Peaks can be identified using a threshold NCC value. The observed mean NCC value offset by the observed standard deviation was found to be an effective threshold choice. Peak positions are indicated by $+$ symbols in Figure 4.6.

An estimate of the period of rotation $p > 0$ is chosen to minimise the total time difference d_{DIF}

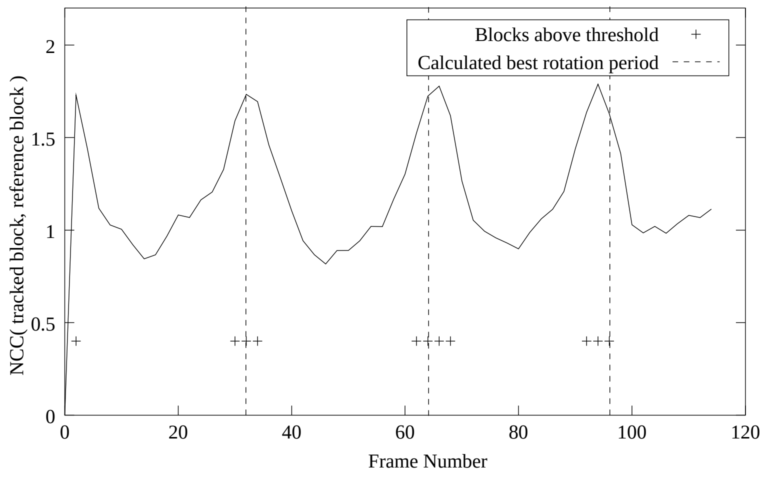


Figure 4.6: *Periodic Similarity results for one cell. Peaks correspond to rotations.*

between each peak position and the closest period step and between each period step and the closest peak position:

$$d_{\mathrm{DIF}} = \sum_{n=0}^{N-1} \min_i\{|t_n - ip|\} + \sum_{i=0}^{\lceil t_{N-1}/p \rceil} \min_n\{|ip - t_n|\} \tag{4.7}$$

The first term ensures that the peak positions are all close to period steps, the second ensures that all period steps are close to peak positions. The resulting best period for the example in Figure 4.6 is indicated by dotted lines.

Alternative spectral estimation techniques such as Fourier Analysis could also be used to estimate the frequency of rotation from the NCC graph. However, unusual components in the graphs due to axes of symmetry and 3D perturbations of the rotation often lead to irregularly shaped curves which can cause problems. Also, for the electrorotation application the speed of rotation is usually not constant, being swept to find when the cell is stationary. Further, for slow rotating cells the number of observed cycles can be very small, and so there can be little data available for the estimation techniques to work with. As the rotation troughs are clearly visible in the temporal domain, it makes sense to estimate the period of rotation directly from these points. The proposed technique performs well and has the benefits of being simple and intuitive.

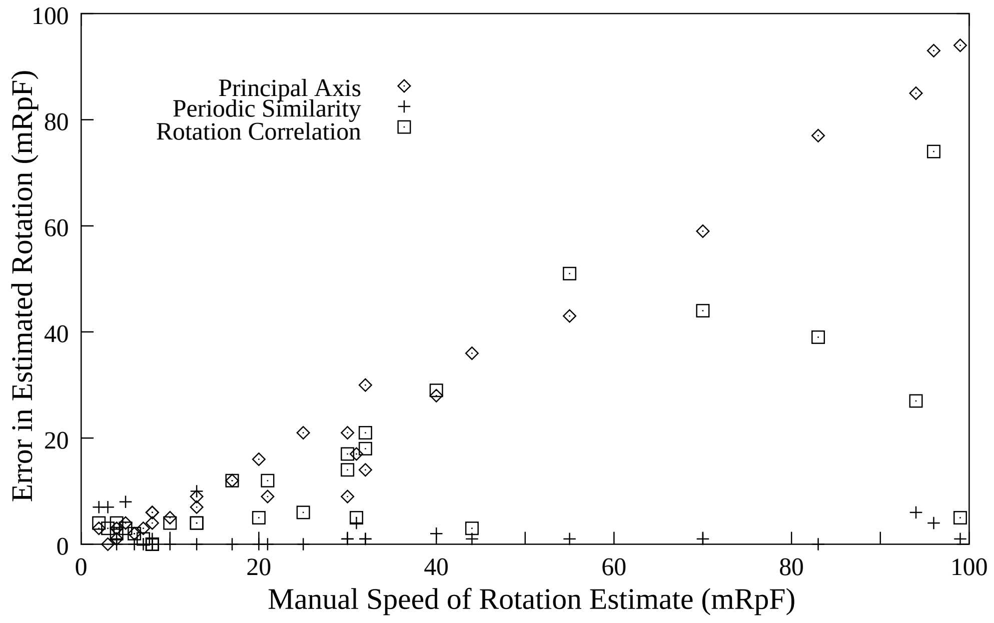


Figure 4.7: *Graph showing mean absolute estimation errors for the Principal Axis, Periodic Similarity and Rotation Correlation methods against the manual estimates (treated as ground truth).*

4.4 Individual Results

Results of applying the three techniques to 30 example data sequences, of lengths from 112 to 881 frames, are given in Table 4.3. In all cases the speed is reported in thousandths of a Rotation per Frame (mRpF). A manual observation of the rotation speed is given as ground truth, this was determined by counting the number of complete rotations in the sequence and dividing by the number of frames taken for these rotations. The precision of the manual estimate is dependent on the image quality, the cell size and on the number of rotations observed. A 16 mRpF rotation of a small cell with radius 10 pixels gives a discernible motion of 1 pixel per frame at the cell edge. For larger cells and/or with observations over multiple rotations it is reasonable to give manual estimates correct to at worst $+/-2mRpF$. Table 4.2 summarises these results with the mean estimate errors. Figure 4.7 shows mean absolute magnitude estimation errors for the Principal Axis, Periodic Similarity and Rotation Correlation methods against the manual estimates.

The Principal Axis method accurately determined the direction of rotation, but gave poor estimates of the magnitude. The method was found to be very sensitive to non-uniform illumination, which biased the principal axis to particular directions, and to noise, which led to poor seg-

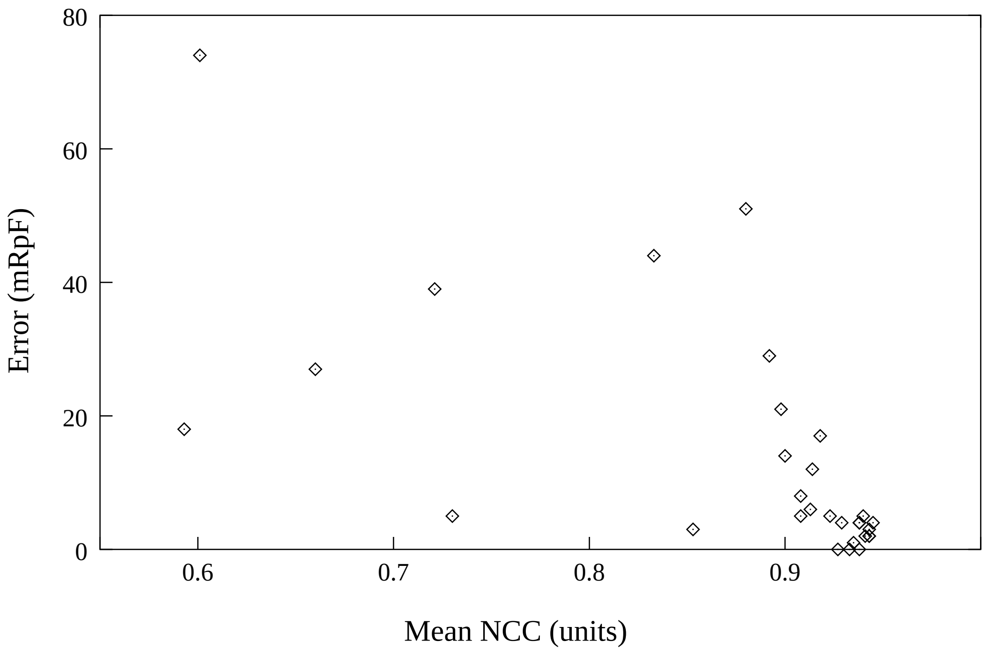


Figure 4.8: *Mean NCC value against average error for the Rotation Correlation method.*

mentation for some frames. From Figure 4.7 it can be seen that the performance deteriorated as the magnitude of the speed of rotation increased.

The Rotation Correlation method accurately determined the direction of rotation and performed well on slow rotation speeds, but poorly on faster rotating cells. The poor performance of the technique on fast rotating cells was due to the large change in appearance of the cell in successive frames. This was a result of both non-uniform lighting conditions and larger perturbations of the cell. In these conditions, the rotated cell reference image appearance differs significantly from the true appearance, and so the method fails to make a correct match. The magnitude of the NCC value can be used to identify such poor correlations. Figure 4.8 shows a scatter plot of the NCC measures against absolute error for the Rotation Correlation method. If estimates with NCC < 0.925 are discarded, then the average error is 2.4mRpF, which is comparable to the Periodic Similarity result (see Table 4.2).

Whereas the Principal Axis method gives a single angular difference between subsequent frames, the Rotation Correlation method gives a relative confidence measure for all the considered angles. This makes the Rotation Correlation method relatively more robust in difficult frames.

In general, the Periodic Similarity approach performed well. However, the technique does

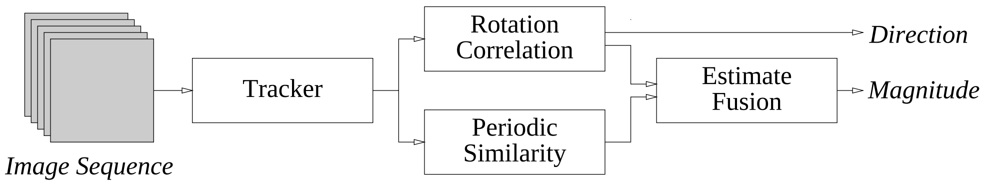


Figure 4.9: *Determining the speed of rotation of cells.*

not calculate a direction of motion and, in a number of sequences, ambiguities in the MAD similarity graphs lead to worse estimates than the Rotation Correlation method.

4.5 Classifier Combination

The relative performance of the techniques at estimating the magnitude of the rotation speed varies considerably over the test cases. Overall, the Periodic Similarity method performs most accurately, but the technique does not give a direction of rotation and in some cases performs worse than the other techniques. The Rotation Correlation method accurately determines the direction of rotation and performs well on slow speeds, but performs poorly on fast speeds of rotation.

It makes sense to use an Information Fusion strategy to combine the absolute speed estimates of these two individual techniques. The poor performance of the Principal axis method suggests that there would be little to gain from including it, and so its estimated are discarded. The complete system is illustrated in Figure 4.9. The magnitude of the fused rotation estimate is a combination of the magnitudes of the Periodic Similarity and Rotation Correlation estimates, while the direction of rotation is just that given by the Rotation Correlation approach. This section describes the information fusion strategy used at the fusion centre.

It is assumed that angles of rotation lie in the range $[-90°, 90°]$ per frame and so the prior probability of estimates outside this range is 0, estimates inside the range have equal prior probabilities. This range covers all speeds in the data set. Limiting the range is a reasonable for the electrorotation problem as, by Nyquist's Theorem, the maximum angle that could be measured is $[-180°, 180°]$ per frame. Since large errors are worse than small errors, the λ_{SE} cost function is chosen. As discussed in Chapter 2 using the SE cost function leads to a fused estimate that is the mean of the posterior distribution.

The posterior distribution $p(r|\mathbf{x})$ is comprised of the individual distributions $p(r|x_{\mathrm{PS}})$ and $p(r|x_{\mathrm{RC}})$, where x_{PS} and x_{RC} are individual estimates of the speed of rotation r from the Periodic Similarity and Rotation Correlation approaches respectively. For simplicity the posterior Periodic Similarity distributions are assumed to be Gaussian with mean x_{PS} (x_{RC}) mRpF and variance 2 mRpF. Following previous authors [84], the individual distributions are combined by averaging.

The results illustrate that the Rotation Correlation technique performs well inside certain ranges and poorly outside those ranges. As illustrated in Figure 4.8, the mean NCC can be used to indicate whether the technique is performing well or poorly. Where the technique is performing poorly, the observation gives no information about the likely speed of rotation and so the posterior distribution is uniform. Such estimates are marked † in Table 4.3. Where the confidence is high, the individual distributions are assumed to be Gaussian with mean x_i mRpF and variance 2 mRpF.

Table 4.3 shows results from applying this combination strategy to the data. The results summary in Table 4.2 shows that the fused strategy in general performs more accurately than any of the individual techniques alone. The fusion approach is successful because it can combine the best estimates of the individual techniques while rejecting any poor estimates from the Rotation Correlation technique. Since the fused estimate is effectively a weighted average of the individual estimates, it is never worse than both individual estimates. However, if the signs of the individual estimate errors are the same, then the fused estimate can be no better than the best individual estimate.

The fused estimate gives only a small accuracy improvement over the Periodic Similarity approach, however, unlike the Periodic Similarity method the fused estimate also incorporates a direction of rotation.

4.6 Summary

This chapter has presented an example in which Information Fusion is applied to a simple Motion Estimation problem where the motion estimates are scalar values rather than motion vectors. The application is determining the speed of rotation of cells undergoing dielectrophoresis, and this chapter has presented a novel method to automatically achieve this. The method has two stages, firstly tracking the cells to remove the translatory component of the motion and

Technique	Average Error mRpF
Principal Axis	23.0
Periodic Similarity	1.9
Rotation Correlation	13.6
Fused Method	1.7

Table 4.2: *Summary of results.*

secondly determining the frequency of rotation from the tracked cell sequence.

The Adaptive Block Matching Algorithm was used to track the cells. An important problem when using this approach is how to update the reference block. This has not received much attention in the past, and so this chapter has presented results from an investigation into different update strategies which concluded that filtering the reference block led to significant improvements in tracking performance.

Three methods for the determining the frequency of rotation were considered. The first is based on a previous solution to the problem, the other two are novel solutions. Although the two novel solutions were found to be more robust at estimating the magnitude of the rotation speed than the previous method, neither performed adequately. By applying an estimation fusion strategy to combine the individual results, a more robust technique was developed. The chosen fusion strategy was based on Bayesian estimation theory, in which the individual estimates were represented by probability distributions.

Cell	Sequence Length (frames)	Principal Axis (mRpF)	Periodic Similarity (mRpF)	Rotation Correlation (mRpF)	Fused Result (mRpF)	Manual (mRpF)
0	118	- 18	31	- 50[†]	- 31	- 32
1	149	- 14	27	- 36[†]	- 27	- 31
2	112	- 9	29	- 44[†]	- 29	- 30
3	319	- 21	29	- 47[†]	- 29	- 30
4	262	- 4	20	- 25[†]	- 20	- 20
5	306	- 4	3	- 17	- 10	- 13
6	303	- 4	8	- 8	- 8	- 8
7	282	- 2	8	- 8	- 8	- 8
8	286	- 2	7	- 8	- 8	- 8
9	426	- 4	7	- 8	- 8	- 7
10	324	- 4	6	- 8	- 7	- 6
11	482	- 3	4	- 6	- 5	- 4
12	410	+ 1	3	- 8	- 6	- 4
13	881	- 5	9	- 6	- 8	- 2
14	848	+ 3	10	+ 6	+ 8	+ 3
15	426	+ 1	13	+ 8	+ 11	+ 5
16	451	+ 5	10	+ 14	+ 12	+ 10
17	112	+ 6	13	+ 17	+ 15	+ 13
18	379	+ 5	17	+ 25	+ 17	+ 17
19	355	+ 12	21	+ 33[†]	+ 21	+ 21
20	188	+ 4	25	+ 31[†]	+ 25	+ 25
21	322	+ 2	31	+ 53[†]	+ 31	+ 32
22	246	+ 12	42	+ 69[†]	+ 42	+ 40
23	162	+ 8	45	+ 47[†]	+ 45	+ 44
24	204	+ 12	56	+ 106[†]	+ 56	+ 55
25	134	+ 11	71	+ 114[†]	+ 71	+ 70
26	135	+ 6	83	+ 44[†]	+ 83	+ 83
27	220	+ 9	100	+ 67[†]	+ 100	+ 94
28	223	+ 3	100	+ 22[†]	+ 100	+ 96
29	181	+ 5	100	+ 94[†]	+ 100	+ 99

Table 4.3: *Table showing automatically and manually calculated frequencies of rotation in thousandths of a Rotation per Frame (mRpF). Entries marked* [†] *have Rotation Correlation mean NCC value* $<= 0.925$. *Where given,* $+$ *and* $-$ *indicate clockwise and anti clockwise rotation respectively.*

Chapter 5
Information Fusion for Accurate and Robust Motion Estimation

5.1 Introduction

The previous chapter described the application of Information Fusion strategies to combine estimates of the speed of rotation of cells undergoing dielectrophoresis. These motion estimates were scalar, unlike the 2 dimensional estimates of the techniques presented in Chapter 2. This chapter considers how Information Fusion strategies may be used to combine the results of the individual 2D Motion Estimation techniques to give more accurate and robust motion estimates in terms of the angular errors, as defined in Chapter 3.

The remainder of this chapter is organised as follows: Section 5.2 describes representations of motion vector estimates and of uncertainties in those estimates which enable Bayesian based Information Fusion to be applied to Motion Estimation. Experimental results from applying the resultant fusion strategy are presented in Section 5.3. Section 5.4 looks at how the choice of cost function affects fusion performance, and suggests a novel cost function to reduce the effects of outliers. Results from applying this new cost function are also presented. A summary of the chapter is given in Section 5.5. The work presented and discussed in this chapter is also reported in [47, 48, 50].

5.2 Representing Motion Estimates for Fusion

As discussed in Section 2.2, Information Fusion has been adapted for a wide variety of applications and the choice of fusion strategy is to a certain extent application dependent. This section discusses how Information Fusion may be applied to Motion Estimation and presents techniques to represent motion vectors and uncertainty in the individual motion estimates.

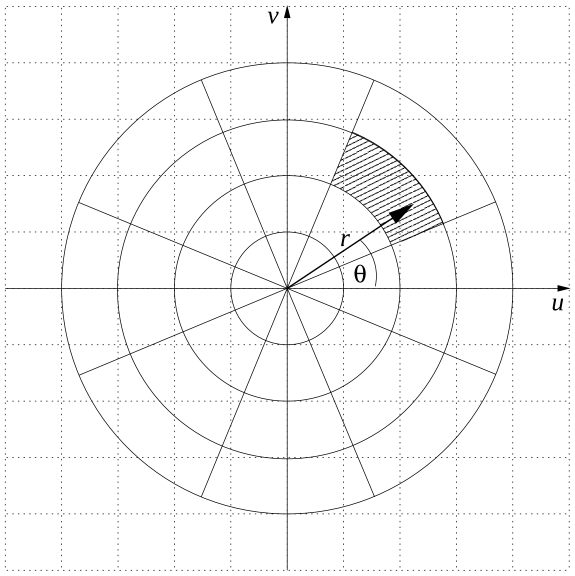


Figure 5.1: *Example quantisation of polar motion vectors to give discrete motion hypotheses which could then be used in a decision fusion strategy. An example motion vector is shown with the associated motion hypothesis shaded.*

5.2.1 Representing Motion Estimates

Many fusion techniques such as committee methods, clustering methods and techniques based on Dempster-Shafer evidence theory work with discrete hypotheses. Although results from correlation based motion estimation techniques such as CAMUS are naturally discrete, in general, motion estimates take values from a continuous range.

Motion estimates could be quantised into a set of hypotheses, this would enable the application of classification fusion strategies. A quantisation based on motion vectors in 2D Cartesian coordinates does not yield an intuitive partitioning of the range. A polar representation of the vectors enables the direction θ and magnitude r of motion to be treated separately, and gives rise to a more natural partitioning by individually quantising the angular direction and distance components. The use of this polar representation for fusing motion estimates is reported in [47, 48]. A typical partitioning and example motion vector are illustrated in Figure 5.1.

This quantisation approach has a number of disadvantages. Firstly, from Figure 5.1 it can be seen that the range of potential motion vectors in each hypothesis and the hypothesis shapes are not uniform. This could be partially rectified by having a non-uniform quantisation of the magnitude component, however the hypotheses would still have non-uniform shapes. Also, as precision requirements increase, the number of hypotheses must increase accordingly. Large numbers of hypotheses can be problematic for classification fusion strategies, especially as the difference between classes becomes negligible. The representational difficulties which arise from enforcing this artificial quantisation suggest that it would be simpler and more elegant to choose fusion strategies which operate directly on the continuous estimates.

As discussed in Chapter 2, stochastic fusion techniques based on Bayes criterion are easily adapted to continuous values $y \in Y$. In this approach, the individual motion vectors $x_0, \ldots, x_{N-1}$ form an observation vector $\mathbf{x}$. The fused estimate $\hat{y}_{\text{FUSED}}(\mathbf{x})$ is made to minimise the risk R associated with the decision system, which is given by Equation(2.15). The risk is minimised when the fused estimate is chosen to minimise the following integral:

$$\hat{y}_{\text{FUSED}}(\mathbf{x}) = \arg\min_{\hat{y}(\mathbf{x})} \int_{-\infty}^{\infty} \lambda(\hat{y}(\mathbf{x}), y) p(y|\mathbf{x}) dy \tag{5.1}$$

Since $\mathbf{x}$ is comprised of the individual motion estimates $x_0, \ldots, x_{N-1}$ with individual distributions $p(y|x_i)$, $p(y|\mathbf{x})$ can be determined if the $p(y|x_i)$ distributions and their inter-dependencies are known. This suggests a representation for the 2D motion estimates $x_i = (u_i, v_i)^T$ using stochastic distributions.

5.2.2 Representing Uncertainty

Where a single distribution shape is used to represent all motion vectors in an image or image sequence, the Central Limit Theorem [153] suggests that it is reasonable to assume that the distribution is Gaussian. As the performance of Motion Estimation techniques is typically data dependent, the accuracy of estimates provided by a particular method is likely to vary over pels in the images and image sequences. It is therefore desirable to use different distributions for the motion estimates from different parts of an image. Assuming that the decisions made by individual techniques are more likely than not to correspond to the true motion, suitable functions should have a maximum at the estimated motion vector and be decreasingly monotonic. Assuming that errors in individual estimations are equally likely in all directions, the distributions would also be expected to be symmetric.

A simple triangular shaped distribution satisfies these requirements, and triangular shaped func-

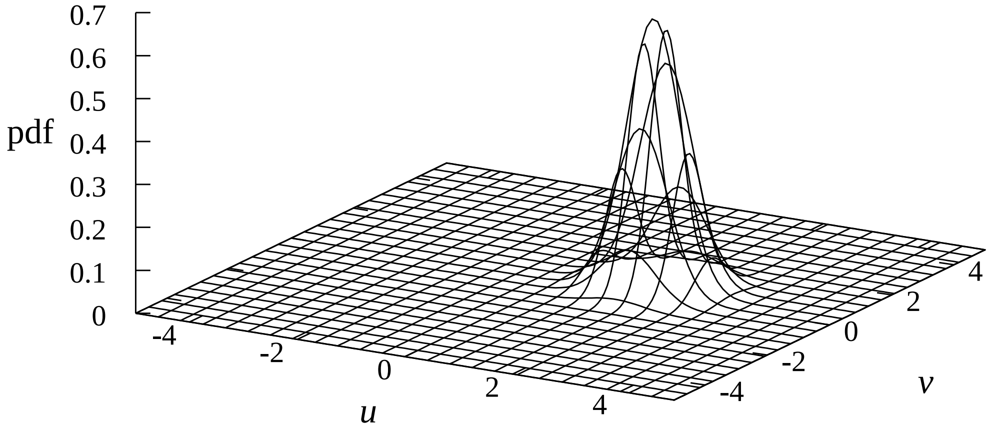


Figure 5.2: *Example probability distribution for a motion estimate* $\mathbf{x} = (u, v)^T = (1.50, 0.50)^T$ *using a Gaussian distribution with element variances* 0.5 *pels and covariances* 0.0 *pels.*

tions can be used to represent motion direction and magnitude estimate uncertainty as separate distributions. However, the direction and magnitude of motion are not typically estimated separately, and so are not independent. A triangular distribution could also be applied in 2 dimensions directly to the motion vector. This function is not a very natural probability distribution, and its derivative is not continuous. A bivariate Gaussian distribution also satisfies the proposed requirements and furthermore, having a continuous derivative simplifies treatment of (5.1):

$$p(\mathbf{a}) = \frac{1}{\det(\Lambda)\sqrt{2\pi}} \exp(-\frac{1}{2}(\mathbf{a} - \mathbf{m})\Lambda^{-1}(\mathbf{a} - \mathbf{m})) \tag{5.2}$$

where Λ is the covariance matrix of the 2D vector $\mathbf{a}$ and $\mathbf{m}$ is the distribution mean. In this thesis it is further assumed that the horizontal and vertical errors are independent and their variances are identical. An example distribution with $\mathbf{m}$ set to be a hypothetical motion estimate $(1.50, 0.50)^T$ and a diagonal covariance matrix with variances 0.5 pels is illustrated in Figure 5.2.

Figure 5.3 shows a histogram of errors for CAMUS applied to the Translating Photo sequence with a scaled Gaussian distribution plotted over the histogram. Clearly the assumption that the distribution is Gaussian does not hold in general. Future work may consider fusion strategies which take account of differences between the shapes of individual distributions. For example,

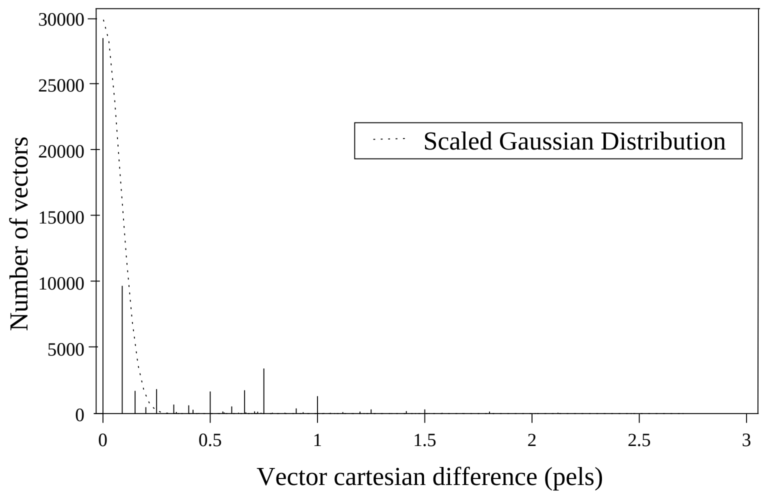


Figure 5.3: *Histogram of CAMUS errors on Translating Photo test sequence.*

the aperture problem would suggest that on edges in the image the distribution would favour motion estimates that lie along edges. However, as the results in Section 3.4.1 suggest, confidence measures for Motion Estimation techniques are not very reliable and so the use of more complex distributions is misleading.

The distribution $p(y|\mathbf{x})$ is determined by the individual $p(y|x_i)$ distributions and their inter-dependencies. Inter-dependencies between different motion estimation techniques may be very complex and varied. Following Kittler *et al.* [84] the distributions will combined by averaging, as this reduces the effects of errors.

5.2.3 Prior Knowledge

An important aspect of the Bayesian approach is the ability to take prior knowledge into account. For example, in many Motion Estimation problems, the no motion hypothesis is most likely. This observation has lead to a number of centre biased motion estimation approaches [119, 121]. However, this assumption does not hold for the test sequences given in this thesis, and so the prior assumption that all motion vectors are equally likely will be adopted.

5.2.4 Section Summary

To summarise, this section has discussed how motion vectors may be represented so that a fusion strategy can be applied. A number of difficulties arise if the motion vectors are quantised, and so 2D Gaussian distributions are to be used to represent the individual motion vectors with the estimates and error in the estimates represented by the means and variances of the distributions. An estimation fusion approach based on Bayes criterion is to be adopted. The individual distributions will then be combined by averaging and a fused estimate chosen to minimise the Bayes' risk, as described in Chapter 2.

5.3 Examples

This section presents the results of applying estimation fusion strategies to combine the results of different motion estimation techniques. The individual motion estimation strategies considered in these experiments are CAMUS, HORN and URAS, as described in Chapter 3. The parameters used for the techniques in these experiments are listed in Table 5.1. The estimation fusion strategies are derived from Bayes estimation theory using the MPE, SE and AD cost functions, as described in Chapter 2. The motion estimates are represented by Gaussian probability distributions as discussed in the previous section. A fourth fusion strategy, which chooses the estimate in which most confidence is placed (MAXCONF), is given as a baseline result.

5.3.1 Implementation Issues

The SE cost function leads to a simple averaging strategy which is easily implemented for 2D motion vectors. However, strategies such as those derived using the MPE and AD cost functions require some evaluation of the probability distributions. This can be computationally expensive.

As the individual distributions are assumed to be unimodal and symmetric about the estimate, when fusing just two results the fused result must lie on a straight line between the individual estimates. This can be used to reduce the computational requirements. In general, these assumptions constrain the fused result to lie in a region defined by the convex hull of the individual estimates, as illustrated in Figure 5.4.

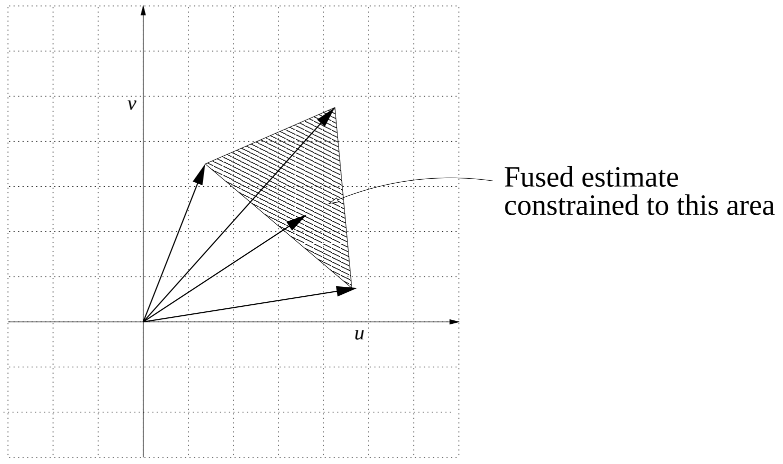


Figure 5.4: *The fused estimate will lie in the convex hull of the individual motion estimates.*

Technique	Parameters
CAMUS	$t = 5$, $w = 3$, $N = 7$
HORN	$\lambda = 100$, max iterations = 5
URAS	temporal smoothing $= 5$ frames, spatial smoothing $= 1$ pel

Table 5.1: *Techniques and respective parameters used in experiments.*

5.3.2 Test Data

In addition to the synthetic Diverging Tree sequence and the real Translating Photo sequence, two more test sequences are used. These are the synthetic Translating Tree and real Advancing Photo sequence, which are illustrated in Figure 5.5 together with their correct motion fields. The Translating Tree sequence is a well known and popular test sequence [107, 136], while the Advancing Photo is a new sequence (recorded for this research and described in Appendix C) which shows a rectangular photograph moving towards the camera at constant speed. As the motion for these sequences is very similar from frame to frame, the accuracy results reported in this chapter are calculated for one frame from the middle of each sequence.

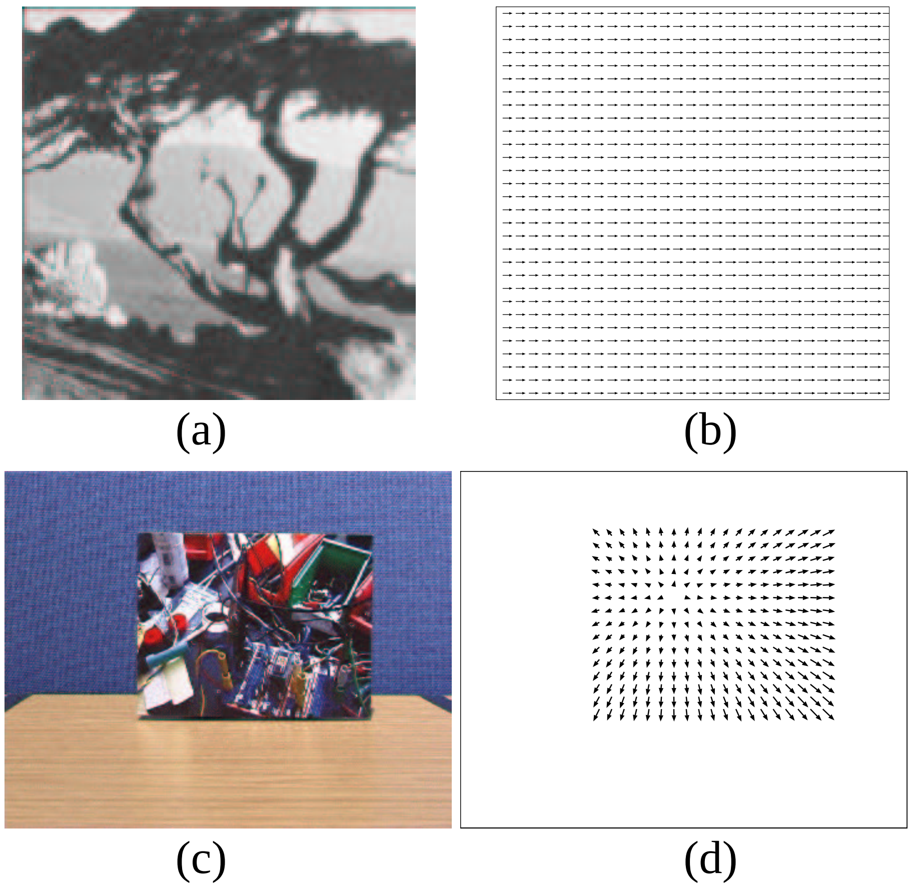


Figure 5.5: *(a) Image from translating tree sequence, (b) correct motion from frame. (c) Image from advancing photo sequence, (d) correct motion for frame.*

	Translating Tree	Diverging Tree	Translating Photo	Advancing Photo
HORN	1.01 (0.27)	0.41 (0.25)	0.36 (0.25)	0.56 (0.27)
URAS	0.81 (0.75)	0.65 (0.50)	0.91 (0.68)	0.90 (0.67)
CAMUS	0.00 (0.00)	0.22 (0.14)	0.42 (0.48)	0.27 (0.31)
CAMUS/HORN				
MAXCONF	**0.00 (0.00)**	0.27 (0.15)	0.35 (0.25)	0.27 (0.31)
Fused MPE	0.01 (0.00)	**0.22 (0.09)**	**0.28 (0.22)**	**0.26 (0.28)**
Fused SE	0.41 (0.11)	0.23 (0.10)	0.42 (0.41)	0.39 (0.27)
Fused AD	0.03 (0.00)	0.23 (0.10)	0.30 (0.23)	0.39 (0.47)
CAMUS/URAS				
MAXCONF	**0.00 (0.00)**	0.23 (0.13)	0.42 (0.48)	**0.27 (0.31)**
Fused MPE	0.02 (0.00)	**0.21 (0.11)**	**0.42 (0.46)**	**0.27 (0.31)**
Fused SE	0.30 (0.28)	0.31 (0.22)	0.89 (0.59)	0.83 (0.55)
Fused AD	0.08 (0.01)	0.27 (0.16)	0.56 (0.82)	0.53 (0.57)
HORN/URAS				
MAXCONF	1.01 (0.27)	0.35 (0.21)	0.36 (0.25)	**0.56 (0.27)**
Fused MPE	**0.78 (0.68)**	**0.33 (0.18)**	**0.35 (0.25)**	**0.56 (0.27)**
Fused SE	**0.78 (0.72)**	0.40 (0.28)	0.89 (0.65)	0.90 (0.52)
Fused AD	0.81 (0.76)	0.37 (0.22)	0.52 (0.68)	0.62 (0.59)
All				
MAXCONF	**0.00 (0.00)**	0.26 (0.14)	0.35 (0.25)	**0.27 (0.31)**
Fused MPE	0.10 (0.17)	**0.19 (0.07)**	**0.33 (0.23)**	**0.27 (0.31)**
Fused SE	0.29 (0.27)	0.28 (0.16)	0.88 (0.53)	0.81 (0.51)
Fused AD	0.37 (0.36)	0.26 (0.14)	0.38 (0.33)	0.38 (0.33)

Table 5.2: *Mean angular errors in radians from applying estimation fusion strategies to combine motion estimates from the HORN, CAMUS and URAS techniques. Error variances are shown in parentheses. For each combination of individual results, the result with the smallest average error is highlighted in* **bold font**.

Technique	Average Angular Error over all test sequences	Average Angular Error without the Translating Tree sequence
HORN	0.59	0.44
URAS	0.82	0.82
CAMUS	0.23	0.30
CAMUS/HORN MAXCONF	0.22	0.30
CAMUS/HORN Fused MPE	**0.19**	**0.25**
CAMUS/HORN Fused SE	0.36	0.35
CAMUS/HORN Fused AD	0.24	0.31
CAMUS/URAS MAXCONF	**0.23**	0.31
CAMUS/URAS Fused MPE	**0.23**	**0.30**
CAMUS/URAS Fused SE	0.58	0.45
CAMUS/URAS Fused AD	0.36	0.68
HORN/URAS MAXCONF	0.57	0.42
HORN/URAS Fused MPE	**0.51**	**0.41**
HORN/URAS Fused SE	0.74	0.73
HORN/URAS Fused AD	0.58	0.50
All MAXCONF	**0.22**	0.29
All Fused MPE	**0.22**	**0.26**
All Fused SE	0.57	0.65
All Fused AD	0.35	0.34

Table 5.3: *Average angular errors in radians over all data sets for different techniques in order of increasing error. For each combination of individual results, the result with the smallest average error is highlighted in* **bold font**. *CAMUS performed very well on the Translating Tree sequence, and it is inappropriate to fuse this result. Average errors without the Translating Tree sequence are given in the last column.*

5.3.3 Results and Discussion

Table 5.2 shows the mean error and error variance of results from applying the different fusion strategies to combine individual results from the HORN, URAS and CAMUS Motion Estimation techniques. Results are given from fusing pairs of strategies and from fusing all three strategies together. A summary of this table showing the average errors over all four data sets is given in Table 5.3. For each combination of individual results, the result with the smallest average error is highlighted in **bold font**.

From Tables 5.3 and 5.4 it can clearly be seen that the MPE fusion strategy matches or outperforms both the baseline strategy MAXCONF and the individual Motion Estimation techniques in almost all cases. The error variances given in Table 5.2 illustrate that there are generally improvements in both mean error and the error variance. This observation suggests that Information Fusion can be used to combine the results of individual Motion Estimation techniques to give more accurate and more robust motion estimates.

The SE cost function gave the poorest performance of the fused results, in some cases leading to greater average errors than even the worst of individual techniques. This is because the SE based strategy does not take account of confidence measures supplied for the individual estimates, thus highly inaccurate estimates are weighted equally with accurate estimates. The AD cost function, which leads to a strategy that takes the median of the distribution, is more robust to inaccurate estimates and so performs better than the SE cost function. However, the AD can still affected by outlier estimates, especially if confidence in these outliers is mistakenly high. Such outlier estimates are common in Motion Estimation data and lead to the AD and SE fusion strategies performing worse than the baseline on average.

Figure 5.6 illustrates graphically the difference in performance between HORN and CAMUS on the 4 test sequences by showing areas where HORN performs best in white and areas where CAMUS performs best in black. Where one strategy performs significantly better than all other strategies, Information Fusion is not appropriate, and in some cases leads to a deterioration in performance. For example, it can be seen from Table 5.2 that CAMUS performs significantly better than either HORN or URAS on the Translating Tree sequence; however the Translating Tree sequence is artificial and such examples are unusual in real world problems. CAMUS performs well in this case because the artificial translating tree sequence is particularly well suited to the assumption of uniform motion over a pel zone. When combining the HORN or

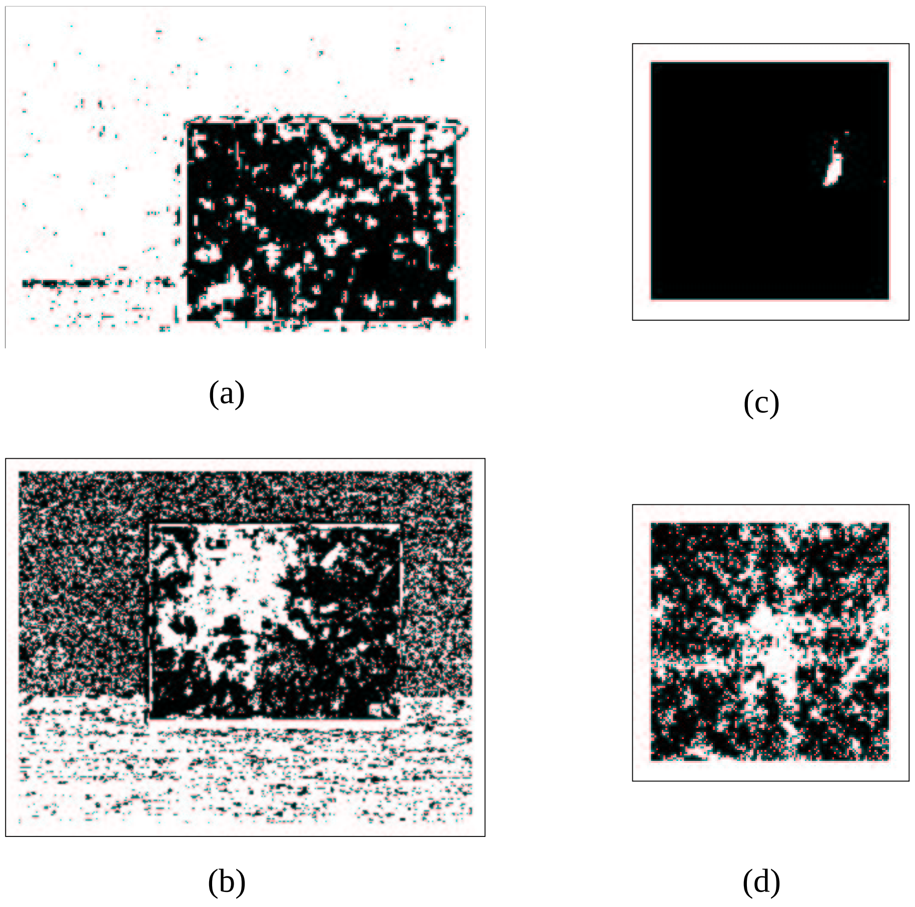


Figure 5.6: *Relative motion errors, black indicates technique CAMUS performed better, white indicates that technique HORN performed better. a) Translating Photo sequence, b) Diverging Photo sequence, c) Translating Tree sequence, d) Diverging Tree Sequence.*

URAS results with the CAMUS on this sequence, errors were introduced. These were the only cases where the MPE based fusion strategy performed worse than one of the individual techniques.

The final column in Table 5.3 shows the average errors without considering the inappropriate fusion of CAMUS results from the Translating Tree sequence. This column makes the benefits that can be gained from applying the MPE fusion strategy even clearer.

Similarly, URAS performs particularly poorly on both the real data sequences. As a result, there is little or no improvement in performance when the URAS results are fused with the CAMUS/HORN results.

From Table 5.2 it can be seen that in some cases the MAXCONF result performs worse than some of the individual results. For example, CAMUS has an average error of 0.22 radians on the Diverging Tree sequence but the MAXCONF strategy has an error of 0.27 radians when fusing CAMUS with HORN and 0.23 when fusing CAMUS with URAS. This is a result of errors in the confidence measures.

5.4 Cost Function Selection

An important problem in Information Fusion is how to combine estimates which are significantly different from each other. This is particularly important when one technique gives an incorrect estimate a high confidence value. This is a common problem in Motion Estimation data and can lead to poor performance of some fusion strategies as discussed in the previous section. This section proposes novel cost functions to improve the robustness of fused motion estimates.

Figure 5.7(a) shows a hypothetical multi-modal posterior distribution for a scalar estimation problem. Such a distribution may arise when the individual distributions from 3 detectors are combined by summing, or when the individual distributions are themselves multi-modal. Figures 5.7(b), 5.7(c) and 5.7(d) are graphs showing the risk associated with each estimate given the MPE, AD and SE cost functions respectively.

As discussed in Section 2.5, the MPE risk is minimum where the posterior probability is maximum. If the peaks in Figure 5.7(a) are interpreted as corresponding to estimates from different detectors, then it can be seen that the MPE strategy is similar to choosing the estimate in which the most confidence is placed. The AD and SE cost functions associate a non zero risk with the other estimates, and so the risk is minimised by estimates somewhere in between the peaks. The SE and AD based risks are minimised by the distribution mean and median respectively.

In this hypothetical case, the peak at the estimate 3 units may be due to one technique giving an incorrect estimate a high confidence value. It is desirable that a data fusion technique should be able to identify and compensate for such outlier estimates.

A hybrid of the AD/SE functions with the MPE can be used to solve this problem. This new cost function assumes that although it is important to have a small error, once the error is above a threshold value Δ the effect of the error is constant. For example, when tracking an object

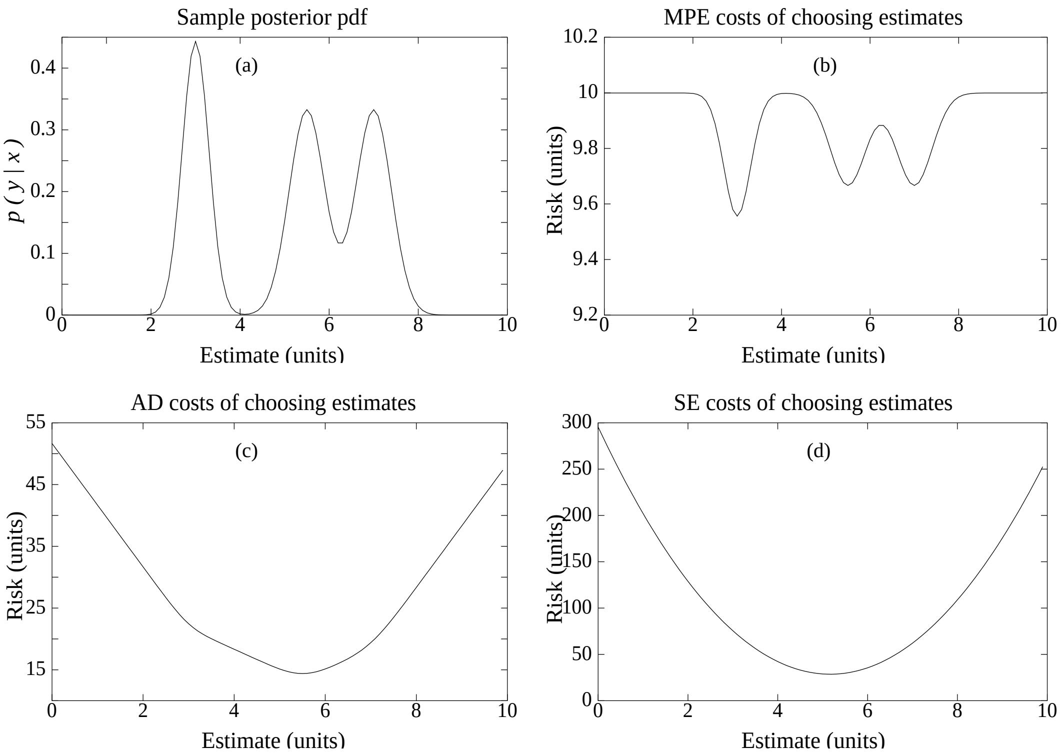


Figure 5.7: *Risks associated with the different hypotheses under the different cost functions.*

then it is important to have as good a position estimate as possible, but any estimates which are so poor as to cause the track to be lost are equally costly, irrespective of the true position. This thresholded cost function is defined as follows:

$$\lambda(a,b) = \begin{cases} \lambda_{\mathrm{C}}(b+\Delta, b) & \text{if } |a-b| > \Delta \\ \lambda_{\mathrm{C}}(a,b) & \text{otherwise} \end{cases} \tag{5.3}$$

Where λ_{C} might be the AD (giving the TAD cost), SE (giving the TSE cost) or an alternative cost function. The use of an absolute difference function is similarly used in robust statistics to minimise the effect of outliers in the data [154, 155]. Substituting the thresholded cost function into the inner integral of (2.15) gives the following fused estimate:

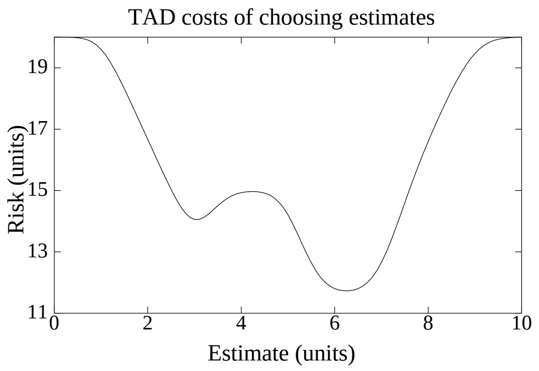


Figure 5.8: *Risks associated with the different hypotheses under the thresholded cost function.*

$$\hat{y}(\mathbf{x}) = \min_{\hat{y}(\mathbf{x})} \left(\lambda_C(\hat{y}(\mathbf{x}), \hat{y}(\mathbf{x}) + \Delta) \left[1 - \int_{\hat{y}(\mathbf{x})-\Delta}^{\hat{y}(\mathbf{x})+\Delta} p(y|\mathbf{x}) dy \right] + \left[\int_{\hat{y}(\mathbf{x})-\Delta}^{\hat{y}(\mathbf{x})+\Delta} \lambda_C(\hat{y}(\mathbf{x}), y) p(y|\mathrm{x}) dy \right] \right) \tag{5.4}$$

The effect of this cost function on the hypothetical pdf of Figure 5.7(a) is illustrated in Figure 5.8, with $\lambda_{\mathrm{C}} = \lambda_{\mathrm{AD}}$ and $\Delta = 2.0$. It can be seen that the outlier estimate has had an insignificant effect on the estimate minimising the risk.

Figure 5.9 presents results from a toy problem where two sensors make observations of the speed of a target moving at a constant velocity of 50mph; inaccuracy in the sensor observations is simulated by two normal distributions with standard deviations 3mph and 5mph respectively. The observations are fused by the MPE, SE and TSE fusion strategies. It is not surprising to observe that the performance of the MPE cost function deteriorates as the accuracy of the confidence measure decreases, while the performance of the SE approach does not. The performance of the TSE approaches that of the MPE with low Δ values and that of the SE with large Δ values. For a range of Δ values the TSE can be seen to outperform both the individual techniques when the confidence errors are small. In general, the choice of Δ will be application dependent. For the motion estimation experiments reported in the remainder of this chapter, Δ is chosen to be 5.0 pel so that errors greater than 5.0 pel are considered equally costly.

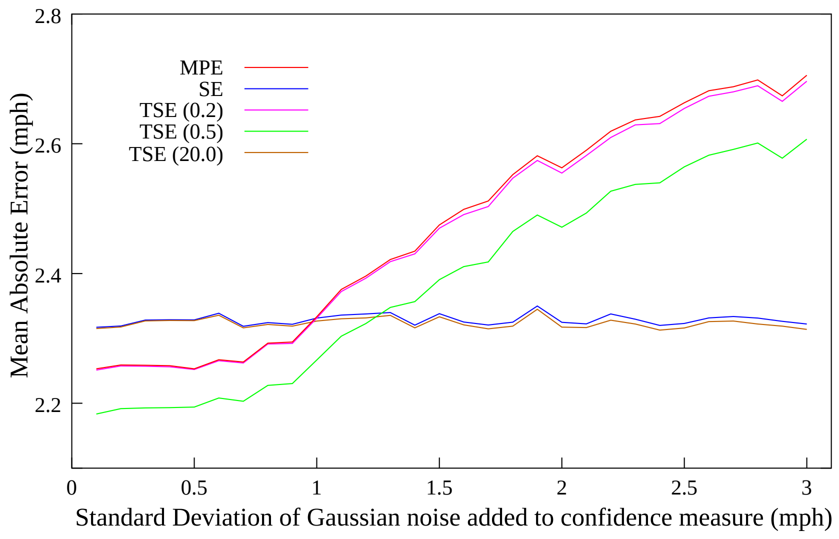


Figure 5.9: *Graph showing the difference in performance between different fusion strategies as noise is added to the individual estimate probability distributions increases.*

5.4.1 Results and Discussion

This section reports results from applying the thresholded cost function presented in the previous section to combine the results of different Motion Estimation algorithms. Again the Diverging Tree, Translating Tree, Diverging Photo and Translating Photo are used as test data. As little improvement was observed from combining all three Motion Estimation techniques, only pairs of techniques are considered here.

Mean angular errors and error variances for the thresholded cost based fusion strategies with $\Delta = 5.0$ pels are shown in Table 5.4. The errors that are less than or equal to the MPE error are shown in bold. It can be seen that in all the example test sequences, the TAD cost function equalled or outperformed the MPE strategy. In most cases the improvement was very small, this is because the MPE strategy was very effective in most cases. However, when fusing CAMUS and HORN on the Translating Photo sequence, the TAD was 10% better than the MPE cost function.

Figure 5.10 illustrates the individual HORN and URAS motion estimates for the Diverging Tree sequence and results achieved by applying data fusion to combine the two techniques using the MPE, SE, AD and TSE and TAD cost functions. Large outlier estimates can be clearly seen in both the HORN and URAS individual results (top row). The result from the SE based strategy, which is independent of the individual distributions, is clearly affected by

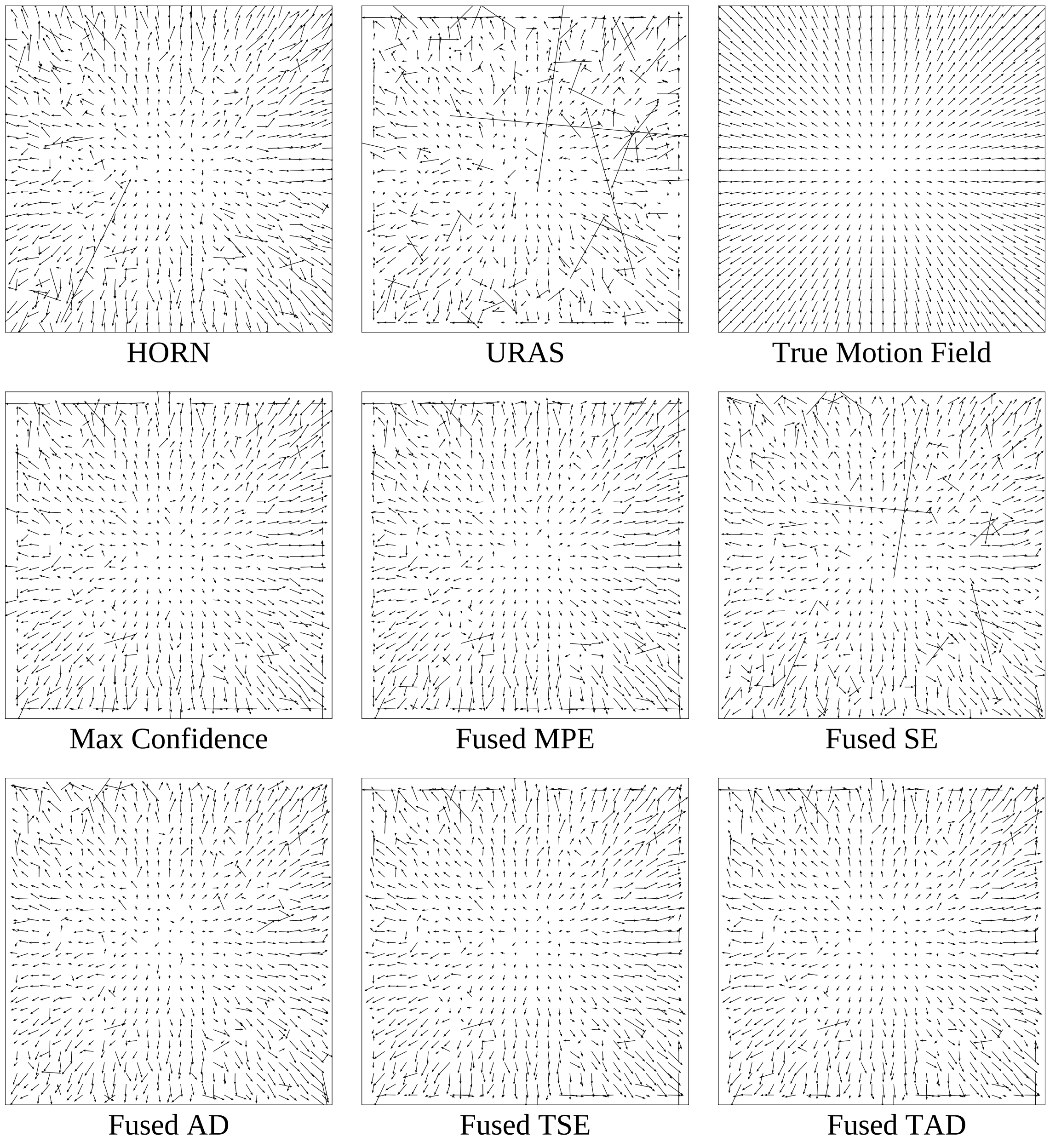


Figure 5.10: *Motion results for the Diverging Tree sequence. Results are shown from two Motion Estimation techniques and from combining the estimates using a variety of fusion strategies.*

	Translating Tree	Diverging Tree	Translating Photo	Advancing Photo
CAMUS/HORN				
Fused TSE	0.01 (0.00)	**0.22 (0.09)**	0.32 (0.23)	0.27 (0.29)
Fused TAD	**0.00 (0.00)**	**0.21 (0.09)**	**0.25 (0.21)**	**0.26 (0.28)**
CAMUS/URAS				
Fused TSE	**0.02 (0.00)**	**0.21 (0.11)**	**0.41 (0.46)**	**0.27 (0.30)**
Fused TAD	**0.02 (0.00)**	**0.21 (0.11)**	**0.42 (0.46)**	**0.27 (0.31)**
HORN/URAS				
Fused TSE	**0.78 (0.56)**	**0.32 (0.18)**	**0.35 (0.25)**	**0.55 (0.26)**
Fused TAD	**0.78 (0.56)**	**0.32 (0.18)**	**0.35 (0.25)**	**0.55 (0.26)**

Table 5.4: *Mean absolute angular errors in radians from applying estimation fusion strategies using the thresholded cost function to Motion Estimation data. Error variances are shown in parentheses.*

these outlier motion estimates. The other fusion strategies, which do take account of confidence measures, are less affected by the outlier estimates and do not have the large erroneous vectors. For the Diverging Tree sequence results illustrated in Figure 5.10, the thresholded TSE and TAD functions performed best, giving a 9% improvement over the baseline MAXCONF.

5.5 Summary

This chapter has considered how Information Fusion may be applied to combine the results of different Motion Estimation techniques. An estimation fusion strategy based on Bayesian estimation theory that can act on continuous values was chosen to overcome limitations which arise if the motion estimates are quantised. Individual motion estimates were represented by 2D Gaussian distributions and combined by averaging. Results from applying the fusion strategy to combine motion estimates from three techniques on four example test sequences showed that in general the fused estimates had smaller average errors than the individual estimates, and also smaller average errors than could be achieved by simply choosing the estimate in which there was the highest confidence.

A novel cost function called the Thresholded Cost function has been introduced. This cost function is useful for problems where, although it is important to have a small error, once the error is above an application dependent threshold value the effect of any larger error is constant. This cost function is also helpful in reducing the effect of outlier peaks in the posterior distribution, which may be caused by large errors in estimates from individual techniques. Outliers are common in the Motion Estimation data set and this cost function was found to give improvements over traditional cost functions.

It is not always appropriate to apply Information Fusion to combine Motion Estimates. In the experiments presented in this chapter, the CAMUS technique was particularly suited to the Translating Tree sequence and significantly outperformed all of the other techniques. As a result, fusing the CAMUS results with other estimates did not lead to any improvements and even resulted in a slight degradation in performance. Similarly, the URAS technique performed significantly worse than the the other techniques on some data sets, and so combining the URAS estimates did not lead to any improvements. Fusing motion estimates is appropriate where the relative performance of the individual techniques is data dependent, and particularly when individual techniques complement each other with one performing well where the other performs poorly. Where appropriate the use of estimation fusion strategies to combine motion estimates can lead to more accurate and robust estimates.

Chapter 6
Information Fusion for More Timely Motion Estimation

6.1 Introduction

The previous chapters have shown that Information Fusion strategies can be used to combine the results of different Motion Estimation techniques to provide more accurate and robust motion estimates than are provided by the individual techniques alone. This thesis proposes that Information Fusion strategies can also be used to provide more timely motion estimates, this chapter describes how this can be achieved.

The remainder of this chapter is organised as follows: Section 6.2 describes the motivations for this work. Section 6.3 describes how Information Fusion techniques can be combined with predictors to solve the AE trade-off problem and so give more timely motion estimates, this is illustrated with both synthetic and real examples. A summary of the chapter is given in Section 6.5. The methods presented and discussed in this chapter have been reported in [51] and [52].

6.2 Motivations

The relative performance of different image processing techniques often varies depending on the image or image sequence being processed. In Motion Estimation, as discussed in Chapter 3, there is often a trade-off between accuracy and efficiency with more accurate techniques tending to require greater computational resources. This trade-off has been characterised by Liu *et al.* [136] as the Accuracy-Efficiency (AE) curve. This is a particularly important issue when processing image sequences. It is desirable to combine the accuracy of slower techniques with the speed of less accurate techniques to give not only more accurate but more timely motion estimates.

The previous chapter has shown that Information Fusion methods can be used to combine the results of different image processing techniques applied to the same frame. However, given that all reasonable image processing techniques take some time to run, in a real time system the estimates they present must always relate to some earlier time point. Different techniques will typically have different computational requirements, and so if run concurrently will give estimates for different time points. The time taken by a given technique can also vary depending on the images being processed. For example, the number of iterations required for the Horn and Schunck technique to converge is not a constant. Some fast search algorithms for block matching techniques have early stopping criteria which can also lead to varying search times (eg. [119, 120]). Thus even when techniques are closely matched in run time requirements, it is not reasonable to expect them to make observations at the same frequency. One solution would be to run all the techniques at the speed of the slowest method, obviously this is not ideal.

In the method proposed by this chapter, illustrated in Figure 6.1, a predictor is associated with each of the individual Motion Estimation techniques to predict the motion at the current time point. Fusion techniques are then used to combine the estimates for that time point. This approach is called the *Predict-Fuse* method, and is described in subsequent sections.

6.3 The Predict-Fuse Method

Since all reasonable image processing techniques take some time to run, an estimate provided by an observer will always relate to some earlier time. Given AE considerations, it can be expected that the more accurate the technique, the more time it will require. It is assumed that techniques can make predictions about future states of the observed process and can give a measure of confidence in these predictions. This is a reasonable assumption if a model can be found to describe activity in the scene.

In this approach, a fusion centre periodically requests from all predictors the estimates for a given time point and the confidence in that estimate. These confidence measures are a function of the expected estimate accuracy and the number of time steps ahead the prediction is for. These predicted estimates and confidences are then combined using the estimation fusion methods described in the previous chapter. The proposed system is illustrated in Figure 6.1. The longer the time between the last observation made by a technique and the predicted estimate, the lower the confidence in the individual estimate, and so the less that it will be valued in determining the fused estimate. In this way, information from all techniques is used when

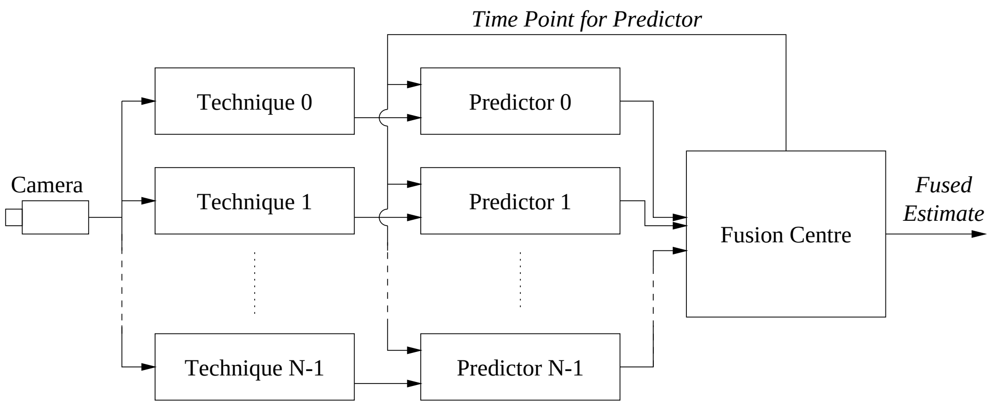


Figure 6.1: *The proposed Predict-Fuse architecture.*

making an estimate for any given time point.

6.3.1 Results - Synthetic Example

To illustrate this idea, this section considers a synthetic example where an object is moving in a straight line at constant velocity away from a point. The distance d_t at time t of the object from its initial position is described by the following equations:

$$d_t = d_{t-1} + T d'_{t-1} \tag{6.1}$$

$$d'_t = d'_{t-1} + U_{t-1} \tag{6.2}$$

where d' is the velocity of the object with $U =\sim N(0, u)$ a random variable representing unknown influences on the process and T the time step. Two processes, A and B, make estimates x of the distance every T_A and T_B time steps respectively. The estimates are simulated by random variables $x_{A,t}$ and $x_{B,t}$ with:

$$x_{A,t} = d_{A,t} + q_A \tag{6.3}$$

$$x_{B,t} = d_{B,t} + q_B \tag{6.4}$$

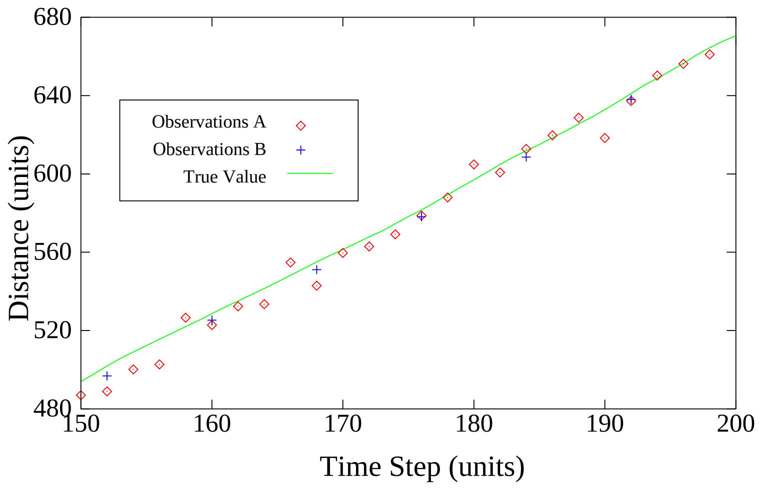


Figure 6.2: *Simulation of a moving object's increasing distance from a point, showing estimates of the distance made by two processes at different frequencies and with different accuracies.*

where q_A and q_B are zero mean normally distributed variables.

The predictor for each observer is a Kalman Filter (see Appendix B.2), which maintains an estimate $\hat{y}$ of the current distance. At each time step, a fusion centre takes an estimate of the current distance and an error estimate from both predictors. The distance and error estimates are provided by the predictor equations of the Kalman Filter [156]. The distance estimates can then be combined using the estimation fusion techniques described in Chapters 2. For this toy problem, equal prior probabilities and the MPE cost function are assumed; the individual results are also assumed to be statistically independent and so the pdf's are combined by taking their product. The individual pdf's are assumed to be normally distributed, with mean and variance given by the distance and error variance estimates from the Kalman filter predictor.

Figure 6.2 shows example estimates and true distances for a simulation with $u = 1$, $T_A = 2$ and $T_B = 8$, the variances of q_A and q_B are 25 and 1 respectively. This simulates a scenario where two different processes are estimating the distance, with the more accurate process requiring more time to run. Under the Predict-Fuse method, fused estimates are made at every time step, which is more frequently than either of the individual techniques.

Figure 6.3 shows the errors (averaged over 300 time steps) for this simulation with $T_B \in [2, 15]$. When T_B is equal to T_A, the average estimate from B is much more accurate than that from A and is preferred so that the fused error approaches that of B. When T_B is larger than T_A, as

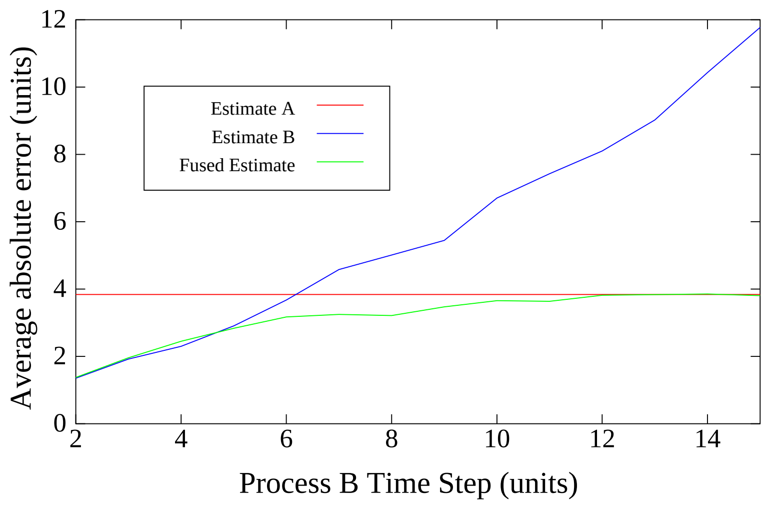


Figure 6.3: *Average absolute errors for the current distance estimates from two different processes and the fused estimate. Graph shows the effect of changing the frequency at which process B makes observations. The fused estimate performs significantly better than either individual estimates for a range of time steps.*

would be expected given AE trade-off considerations, the predictions have to be over a longer time span and so A gives the better estimate. There is a range of T_B values in which the fused estimate outperforms both individual estimates. In this range, the fused estimate is made more frequently and more accurately than either of the individual techniques.

6.3.2 Results - Motion Estimation

The synthetic example given in the previous section is a relatively simple problem, in that the underlying probability distributions and processes are known. This section shows that the technique can also be successfully applied to Motion Estimation. The intention is to combine techniques from different parts of AE space to create a system which sits closer to the origin, where the time axis is now a measure of observation frequency.

Motion Estimation has many applications and the choice of model for the predictors is application dependent. For simplicity and to reduce the computational overhead of the predictors, it was assumed for this example that the optical flow is comprised of points (the pels) moving independently and with constant motion. So the motion predictors simply calculate the expected location for each pel in T time steps and predict that the motion at that location will be the current motion of the pel. This is a reasonable model of the motion in the Translating Photo

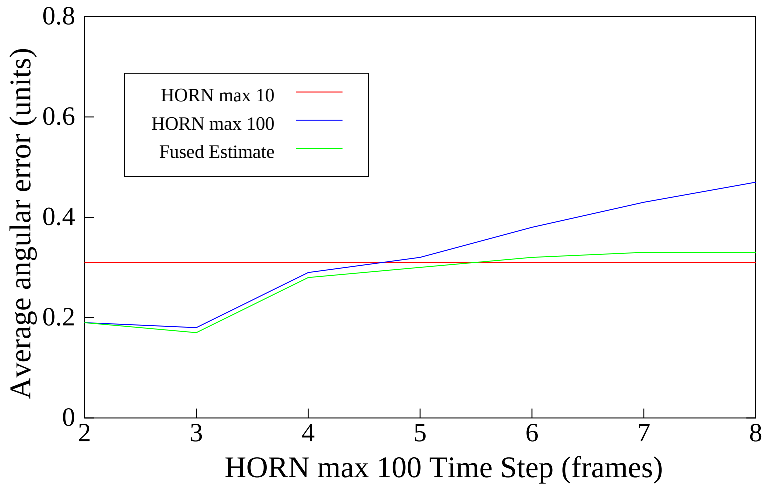


Figure 6.4: *Average angular errors for motion estimates from HORN with max 10 and 100 iterations and the fused estimate. Graph shows the effect of changing the frequency at which HORN 100 makes observations. The fused estimate performs significantly better than either individual estimates for a range of time steps.*

sequence. A similar model is used by Singh [157] as the transition function in his Kalman filter approach to updating motion estimates. Singh also uses the Kalman filter to provide confidence measures, however the additional spatial smoothness model imposed makes the filter unsuitable for this fusion application as the model may conflict with or favour particular techniques.

As discussed in Chapters 3 and 5, it is difficult to find appropriate confidence measures and probability distributions to accurately portray the motion estimates. As in Chapter 5, uncertainty in the individual motion estimates is represented by 2D Gaussian probability distributions whose means and variances are the estimated motion vector $(u, v)^{\mathrm{T}}$ and confidence measure respectively. Thus a large variance indicates low confidence. For this example, equal a priori probabilities and the MPE cost function are assumed. The individual techniques are combined by averaging. The falling confidence in long range predictions was implemented by assuming that the error variance increases by a factor of the time step T.

The Predict-Fuse method can be applied to combine two versions of the same technique from different parts of the AE-curve. As discussed in Chapter 3, the maximum iterations parameter of the differential HORN technique can be varied to give an AE curve. As the computational requirements of HORN with 100 iterations are greater than with 10 iterations, it can be expected that HORN 100 will require more time to process observations and so will make observations with less frequency.

Figure 6.4 shows a graph of the mean angular error over 30 frames of the Translating Photo sequence for the predict estimates from HORN with 10 and 100 iterations and for the fused result. HORN with 10 iterations is assumed to make observations every 2 frames. The horizontal axis gives observation frequencies for HORN 100 which vary from every 2 frames to every 8 frames. As the frequency of observations for HORN 100 decreases, so the error in the motion estimates increases. At high frequencies HORN 100 performs better than HORN 10, but at lower frequencies HORN 10 performs best.

The shape of the graph in Figure 6.4 is similar to the synthetic example in Figure 6.3, and there is a range in which the fused result outperforms the individual results. The fused result performs slightly worse than the better of the individual results at the extreme ends of the graph. This is due to errors in the fusion because of inaccuracies in the confidence measures.

Where a slow technique takes a relatively very long time to make estimates, results from the faster technique may be far more accurate. This may result in a situation where there is nothing to be gained by including the results from the slower technique and Fusion is inappropriate.

Figure 6.5 illustrates the motion estimates from the two individual predictors and the fusion centre corresponding to the experimental results in Figure 6.4 with HORN 100 observations every 5 frames. Figure 6.6 shows results from the same techniques applied to the Hamburg taxi sequence. Although a quantitative evaluation of the performance is not possible as ground truth is not known, qualitatively, it can be seen that the fused estimate better captures the motion of the three vehicles and pedestrian in the scene than either individual estimates. Both the individual techniques have a number of mistaken motion estimates where there is no motion in the scene. There are fewer such errors in the fused estimate.

The Predict-Fuse method can also be applied to combine results from different techniques. Table 6.1 gives the mean angular errors from combining HORN with max 100 iterations and CAMUS with block size 5×5 pels, search radius 3 pels and time range 5 frames applied to the translating photo sequence. The error for HORN is smaller than for CAMUS so using the AE trade-off argument it was assumed that HORN had greater computational requirements than CAMUS and made observations every 5 frames while CAMUS made observations every 2 frames. In reality, the relative run-time of these techniques would be platform dependent. The fused estimate, made at every frame, is significantly better than either of the individual results. Much of this improvement is due to combining two techniques which complement each other.

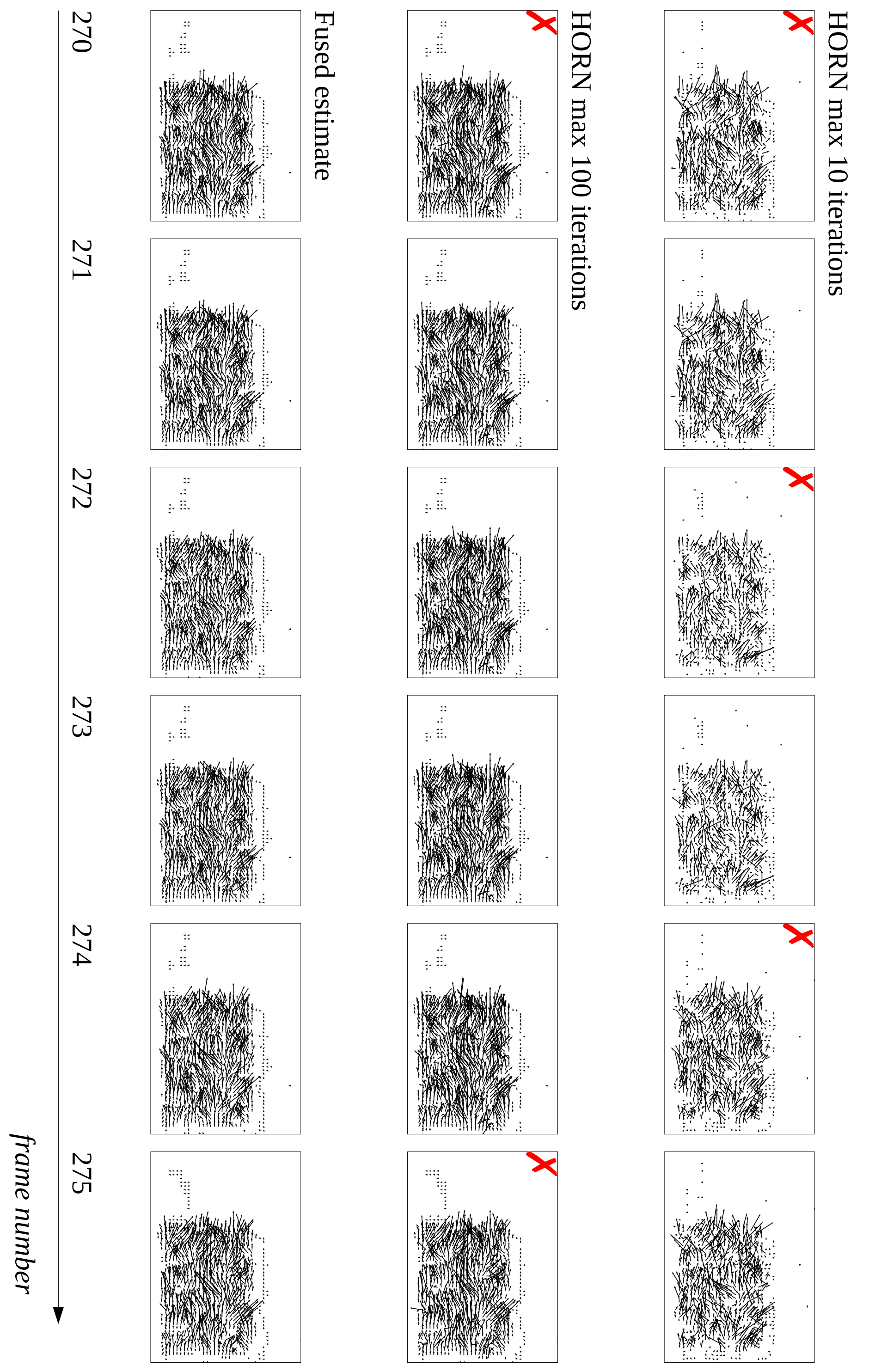


Figure 6.5: *Motion estimates for HORN with 10 and 100 iterations and the fused result on the Translating photo sequence. Observations are made by HORN 10 every 2 frames and by HORN 100 every 5 frames. A cross indicates estimates from the new observations.*

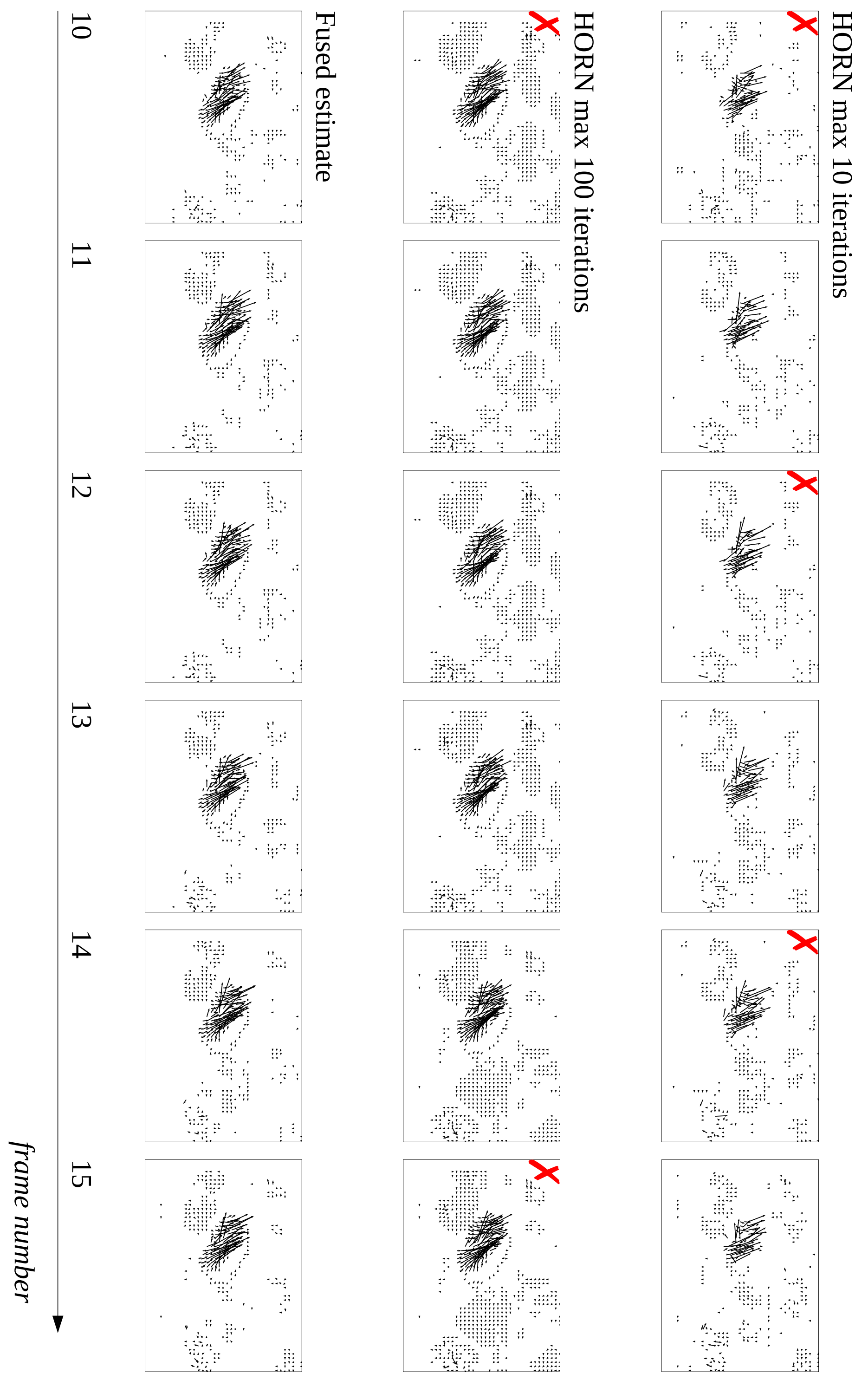


Figure 6.6: *Motion estimates for HORN with 10 and 100 iterations and the fused result on the Hamburg Taxi sequence. Observations are made by HORN 10 every 2 frames and by HORN 100 every 5 frames. A cross indicates estimates from the new observations.*

Technique	Mean Angular Error radians
CAMUS	0.62 (0.56)
HORN	0.38 (0.31)
FUSED	0.25 (0.24)

Table 6.1: *Mean angular errors (variances in parentheses) from combining HORN with max 100 iterations and CAMUS with block size 5×5 pels, search radius 3 pels and time range 5 frames applied to the translating photo sequence.*

Figure 6.7 illustrates the individual and fused motion vectors over 6 frames of the Translating Photo sequence resulting from fusing the HORN and CAMUS estimates. Clearly CAMUS performed better on the foreground than HORN, but worse on the background. The fused estimate better captures the motion in the scene than either of the individual estimates. Over the whole image, the fused estimate has a significantly lower error. However, close inspection of Figure 6.7 shows that some of the fused estimates on the foreground are less accurate than estimates from CAMUS. This is due to the difficulty in finding appropriate confidence measures for the individual techniques.

Having some techniques make more frequent observations than others has an important consequence in that the techniques will be better able to detect motion at different speeds. For example, a very slow moving object may appear to be stationary to a technique making frequent observations, and the motion will be more easily detected by a technique making less frequent observations. This means that where motion is occurring at different speeds in an image sequence, systems such as the Predict-Fuse method that allow techniques to run at different speeds can have an advantage.

6.4 Implementation Issues

The ideas presented in this chapter and in previous chapter centre around an inherently parallel fusion architecture. A detailed discussion of parallel architectures is beyond the scope of this thesis, however this section briefly describes how the architecture can be implemented using the popular Message Passing Interface (MPI) standard.

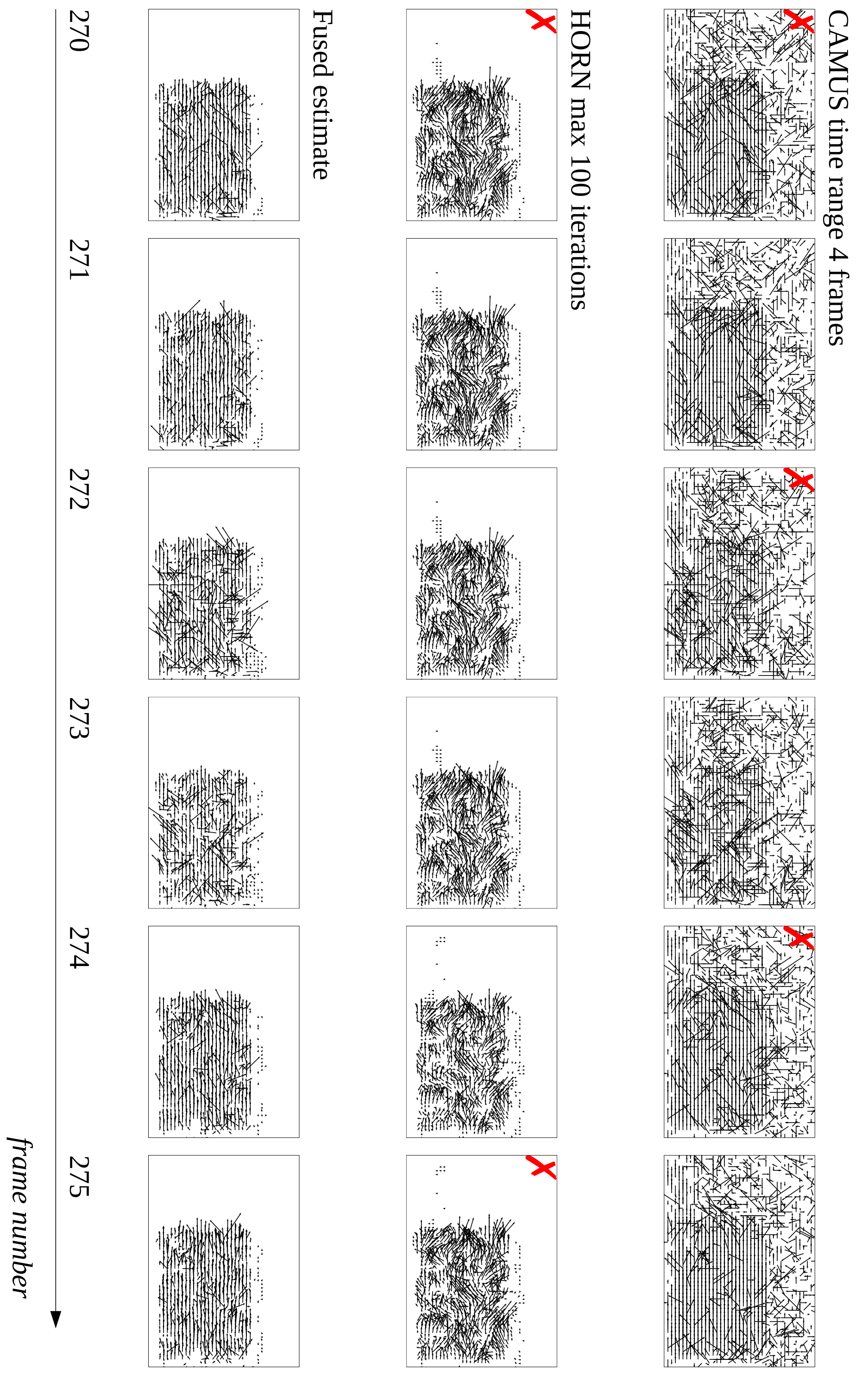


Figure 6.7: *Motion estimates for HORN with 100 iterations, CAMUS with* 5×5 *pel block size, search radius* 3 *pels and time range* 5 *pels, and the fused result on the Translating Photo sequence. Observations are made by CAMUS every 2 frames and by HORN every 5 frames. A cross indicates estimates from the new observations.*

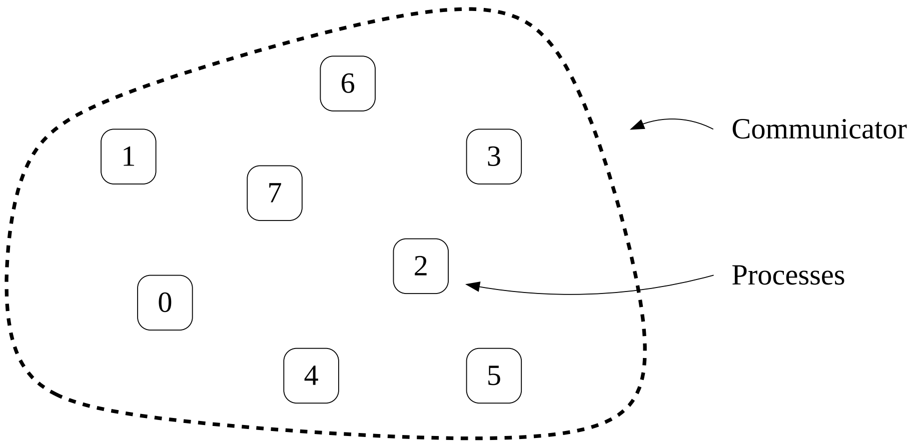


Figure 6.8: *In the Message Passing Interface (MPI), N processes make up a* communicator. *The processes run autonomously and are uniquely identified by their* rank.

6.4.1 Brief Overview of MPI

The MPI standard [158] provides a framework for parallel programming that enables message passing between different processes which may be running on different processors. The standard was designed to be portable and enables efficient implementation of parallel programs over a range of architectures. MPI is supplied as a library of routines. C, C++ and Fortran implementations of the standard are freely available including MPICH [159] which is available from the Mathematical, Information and Computer Sciences Division of the US department of Energy and CHIMP-MPI [160] which is available from the Edinburgh Parallel Computing Centre.

In an MPI program, as illustrated in Figure 6.8, multiple processes, each uniquely identified by an integer *rank*, run autonomously. Processes are grouped together in a *communicator* and can exchange messages with other processes in the communicator using functions from the MPI library.

Communications can be *Collective*, where a group of processes exchange messages, or *Point-to-Point* where one process sends a message to one other process. Communications modes are described in terms of how they are sent, but for a message to be received the recipient process must call a “Receive Message” function. There are 4 communication modes: Standard Send, Synchronous Send, Buffered Send and Ready Send. These modes differ in how their completion relies on receipt of the message. Standard Send completes once the message has been sent into the communications network. This may or may not imply that the message has been re-

MPI Datatype	C Datatype
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double

Table 6.2: *MPI and corresponding C datatypes.*

ceived, depending on the communication architecture. Synchronous sends complete when the message has been received. Buffered sends complete immediately, the message is stored in a system buffer to be transmitted when possible. Ready sends also complete immediately, but are only guaranteed to succeed if a matching receive has been posted, otherwise the behaviour is undefined.

All communications modes can act either in *blocking* or *non-blocking* mode. In non-blocking mode, a process can carry on with any processing that does not affect the send buffer while the message is sent while in blocking mode the process must wait until send has completed. Non-blocking communications have an associated function to test for call completion. There is no advantage to having non-blocking Buffered or Ready Send communications. Receive calls can also be blocking or non-blocking.

Messages are arrays of elements which correspond to a particular data type. A number of data types are defined which correspond to standard C and Fortran data types, including MPI_CHAR (corresponding to a C char) and MPI_INTEGER (corresponding to a Fortran INTEGER). A special data type MPI_PACKED enables representation of structures. A list of MPI datatypes and their corresponding C datatypes is given in Table 6.2 [158].

6.4.2 Predict-Fuse Implementation

The Predict-Fuse architecture illustrated in Figure 6.1 may be implemented with the individual techniques, predictors and the fusion centre running as parallel processes, possibly on different processors. The experiments reported in this thesis have not been run in real-time, and so image capture has been simulated by reading from disk. In a real-time architecture, a process may also be used at the camera as a server to provide the individual techniques with images as required. Figure 6.9 illustrates how the Predict-Fuse architecture was implemented on the departmental Sun network using MPI.

The Predict-Fuse architecture contains both synchronous and asynchronous components. The processing of images by the individual techniques is asynchronous, while synchronous outputs from the predictors are required by the fusion centre. Table 6.3 provides a key to Figure 6.9 illustrating how the communication was implemented using functions from the MPI standard. Buffered Sends are used for asynchronous components of the architecture while buffered Synchronous Sends are used for synchronous components.

The architecture assumes that the predictor processes are faster than the individual techniques. This is a necessary assumption for the Predict-Fuse architecture to be of practical use.

The messages being passed in this implementation are frames and motion estimates from left to right (communications 2,3 and 5), and requests for data from right to left (communications 1 and 4). For 256 greyscale images, frames can be passed as arrays of MPI_CHARs or MPI_BYTEs. Motion estimates can be passed as arrays of MPI_DOUBLEs with alternating horizontal and vertical motion components, or MPI_INTs if the motion vectors are appropriately scaled and rounded. Communications that request frames and motion predictions require only MPI_INTs.

6.5 Summary

This chapter has considered the problem of information fusion where observations of a process are processed by different techniques at different speeds and with different accuracies. This is a typical situation in Motion Estimation characterised by the Accuracy-Efficiency curve. It is desirable to combine the speed of the less accurate techniques with the accuracy of the slower ones, while taking account of the fact that estimates from the individual techniques may relate

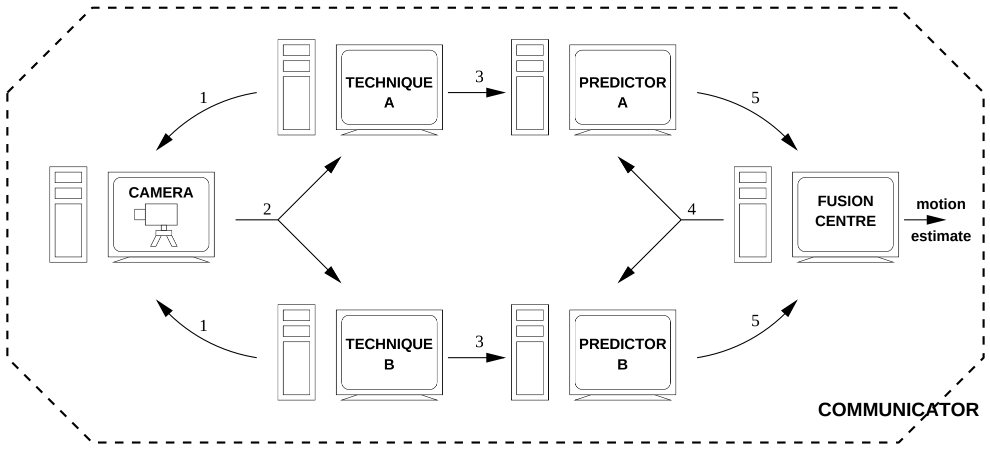


Figure 6.9: *Architecture for an MPI implementation of the proposed Predict-Fuse architecture showing communication between the different components of the architecture. A key to the communication components is given in Table 6.3*

Label	Communication Type	Description
1	Blocking Point-to-Point Synchronous Send with Non-Blocking Receive	Request frame for processing
2	Non-Blocking Collective Synchronous Send with Blocking Receive	Send frames to all techniques that have requested them.
3	Non-Blocking Point-to-Point Buffered Send with Blocking Receive	Deliver frame to predictor
4	Blocking Collective Synchronous Send with Non-Blocking Receive	Request predicted Motion Estimates for time T.
5	Non-Blocking Point-to-Point with Blocking Receive	Provide estimate for the Fusion Centre.

Table 6.3: *Key to communication types for the MPI implementation of the Predict-Fuse architecture in Figure 6.9.*

to different and previous time points. A method has been presented which uses predictors to provide estimates for the individual techniques at the current time point, then combines the estimates using Information Fusion methods.

Results from applying the Predict-Fuse method to a synthetic problem have illustrated that valuable improvements may be made using this information fusion strategy, which enables more timely estimates (in terms of estimate frequency) than can be achieved from any single technique alone. Improvements are possible when the difference in computational requirements and performance lies inside a (problem dependent) range. The performance of Motion Estimation techniques, in terms of accuracy and computational requirements, can be both data and platform dependent. However, results presented in this chapter show that the Predict-Fuse method can be applied to combine the results of different Motion Estimation techniques to get more timely estimates.

The techniques described in this and previous chapters have inherent parallelism. This chapter has also provided a brief description of how the Predict-Fuse architecture has been implemented using the Message Passing Interface (MPI).

Chapter 7
Summary and Conclusions

7.1 Introduction

This thesis has considered the hypothesis that Information Fusion can be applied to combine the results from different Motion Estimation techniques to give more accurate, robust and timely motion estimates. Appropriate fusion strategies for combining motion estimates have been identified and expanded, and novel techniques have been described which facilitate the application of Information Fusion to Motion Estimation.

The remainder of this chapter is organised as follows: Section 7.2 summarises the thesis content and identifies the main contributions. Section 7.3 draws conclusions from the work presented in this thesis. Section 7.4 discusses possible topics for future work and some final comments are made in Section 7.5.

7.2 Summary

The theme of this thesis has been the application of Information Fusion strategies to Motion Estimation. Information Fusion is applied to a wide range of applications, and the choice of fusion strategy is application dependent. Chapter 2 discussed a number of application examples and identified from these a variety of fusion strategies. Fusion strategies based on Bayesian estimation theory were considered particularly suitable for Motion Estimation because confidence in individual estimates can be taken into account and because the strategies can work with continuous valued estimates. Strategies based on Bayesian decision theory have also been compared favourably to alternative strategies in the literature.

An alternative strategy to Bayesian estimation theory is Fuzzy Set theory. A important problem when adopting a Fuzzy approach to Information Fusion is the choice of appropriate fuzzification, aggregation and defuzzification operators. Although a Fuzzy approach was not adopted for this thesis, Chapter 2 also drew an analogy between the Fuzzy and Bayesian approaches that may help in the choice of these functions.

Chapter 3 presented an overview of Image Processing techniques for Motion Analysis. Techniques were classified as being primarily designed for Motion Compensation, where the aim is to compress image sequences, or for Motion Estimation, where the aim is to describe the real world causes of image intensity changes. A novel technique was proposed for making fast though rough estimates of the direction of motion from differences in the DCT coefficients of successive images. This work may have application to reducing the search space in fast implementations of the Block Matching algorithm. Examples of key Motion Estimation approaches (differential and correlation based methods) were introduced and their performance investigated using both synthetic and real test sequences.

An illustrative estimation fusion example was given in Chapter 4, where estimates of the speed of rotation of cells undergoing electrorotation were combined using a fusion strategy based on Bayesian Estimation theory. The fused rotation speed estimates were shown to be more accurate than the individual estimates alone. This work is important for the automation and hence commercialisation of dielectrophoresis methods.

The electrorotation problem described in Chapter 4 used the Block Matching Algorithm to track cells so as to remove the translatory component of their motion. Reference block updating strategies for the tracking algorithm were investigated, and the use of filters was found to lead to significant improvements in tracking performance. Further work by other researchers [161] has since investigated the robustness of the proposed strategies to variations in their parameters.

The problem of combining 2D motion estimates was considered in Chapter 5. It was first proposed that motion estimates might be quantised into discrete motion hypotheses to enable the application of classification fusion strategies. However, this artificial quantisation raised a number of questions and was not found to be a natural or intuitive representation for the problem. Instead, estimation theory was applied to combine the continuous motion estimates directly.

Chapter 5 presented results from combining three Motion Estimation techniques. The techniques were chosen to be representative of the two main Motion Estimation approaches: differential and correlation based. It was found that in general just two of the techniques were required to get the best results. There exist many Motion Estimation strategies and the relative performance of these strategies is data dependent, however, due to the computational requirements of Motion Estimation strategies, it is unlikely that many more strategies would ever be combined in a single system.

The issue of cost function selection for estimation fusion was considered, and a novel cost function presented which takes account of the fact that, although it is important to have a small error, once the error is greater than some threshold the effects of any further increase remain constant. This cost function also improves the robustness of the motion estimates to outliers. Chapter 5 presented results from real and synthetic test data that showed that Information Fusion can be used to combine motion estimates to give more accurate and robust motion estimates in terms of angular errors.

There is often a trade-off between the accuracy of Motion Estimation techniques and their computational requirements, with more accurate techniques tending to have greater computational complexity. Chapter 6 showed that by fusing predictions based on estimates from fast but less accurate techniques with predictions from slower but more accurate techniques it is possible to get more accurate motion estimates more quickly. Chapter 6 also described an implementation of the fusion architecture in parallel using the Message Passing Interface standard.

7.3 Conclusions

This thesis proposed that Information Fusion can be used to combine the results of different Motion Estimation techniques to give more accurate, robust and timely estimates. There exist a variety of fusion strategies that might be adopted to combine motion estimates. From the discussions presented in Chapter 2, this thesis concludes that Bayesian Estimation theory provides an effective and appropriate framework for combining motion estimates.

From the results presented in Chapter 5, this thesis concludes that where one technique is found to consistently perform better than the alternatives it is not appropriate to apply fusion strategies. However, where the relative performance of different techniques varies, and especially where the individual performances are complementary, improvements in the accuracy and robustness of motion estimates can be obtained by fusing the individual estimates. In one example of a real image sequence where the individual performances of two different Motion Estimation techniques were complementary, fusing the individual estimates was shown to give a very significant 30% improvement in accuracy over the better of the individual results. Where fusion is not appropriate due to one technique being far more accurate and robust than the alternatives, the fused motion estimates will be similar to those from the better of the individual techniques.

Motion Estimation has an inherent temporal aspect and so the timeliness of techniques, in terms of the frequency at which estimates can be made, is important. There is often a trade-off between the speed and accuracy of Motion Estimation techniques. From the results presented in Chapter 6, this thesis concludes that motion estimates from slower but more accurate techniques can be effectively combined with estimates from faster but less accurate techniques to give more frequent and more accurate estimates.

In the course of this thesis, the choice of reference block updating strategy for tracking with the Block Matching Algorithm has also been investigated. From these investigations, this thesis concludes that the performance of the Block Matching Algorithm is very dependent on the choice of update strategy and that appropriate choice of update strategy can lead to significant improvements in tracking accuracy. The experimental results presented in Chapter 4 suggest that the use of a Kalman filter to update the reference block is a very robust strategy which incurs relatively small computational overhead.

7.4 Future Work

The scope of this thesis is limited in a number of ways which might be developed in future work. Firstly, only the fusion of 1D and 2D motion estimates is considered. This is not a severe limitation as the majority of research and applications of Motion Estimation work with 2D vector fields, and extending the techniques developed here to 3D vectors is straight forwards.

Estimates derived from applying Motion Estimation algorithms to a single sensor only are considered, this avoids the need to co-register the results. This problem was considered outside the scope of the thesis. There are a number of possible advantages to fusing motion estimates from different sensors including benefits due to having better views of the target and having differences in sensor noise.

The fusion strategies presented assume simple (Gaussian) probability distributions for the individual estimates. These were chosen to simplify implementations and because the lack of accurate confidence measures makes the use of more complex distributions at best misleading. However, considerations such as the aperture problem suggest that more specific distributions might be appropriate, and these could be considered in future work. Furthermore, it was assumed that the results from individual techniques are independent. Again, this assumption was made because it is not straight forward to find reliable correlations between the individual techniques and because the independence assumption simplifies the fusion framework by enabling individual techniques to be treated separately. However, if two techniques are based on similar underlying assumptions then some correlation between their results would be expected, this might also be considered in future work.

Chapter 5 presented a novel cost function for Estimation Fusion. This cost function was a hybrid of the Minimum Probability of Error and the Squared Error or Absolute Difference cost functions. This novel cost function takes a threshold parameter. Although results from varying this parameter were presented for a toy problem, the effect of this parameter on fusing motion estimates was not investigated empirically. This was not done because the novel cost function was presented in the context of a subjective criterion, namely that although it is important to have a small error, once the error is above a threshold value the error is constant. Further investigation of this novel cost function would not change the conclusions of this thesis, but is of interest so may be considered in future work.

The estimation fusion strategies explored in this thesis have other applications outside Motion Estimation. Work has already considered how the techniques explored in this thesis can be applied to combine the results of Artificial Neural Networks, and future work may consider other application areas.

The investigation of Block Matching Algorithm update strategies presented in this thesis did not consider the Block Matching Algorithm in the context of a system that filters the position estimates. As most practical systems filter position estimates, this would also be an appropriate subject for further investigation.

7.5 Final Remarks

Motion Estimation is an important research area with a number of applications. As a result, there exist a wide variety of techniques for Motion Estimation. The relative performance of these techniques is data dependent. Where the performances of different Motion Estimation techniques are complementary, Information Fusion strategies can be used to combine the individual estimates to give more accurate, robust and timely estimates. This work is likely to be of most benefit to Motion Estimation applications that must operate under a wide variety of conditions, and so are likely to provide situations in which the relative performance of different Motion Estimation techniques varies. For example, this work is likely to be of benefit to surveillance applications which are expected to operate in a range of locations including inside buildings, overlooking car parks and on natural outdoors scenes. Outdoors surveillance applications must also deal with weather changes which will give rise to a range of visibility conditions.

References

[1] T. Camus, “Real-time quantized optical flow,” in *IEEE Conference on Computer Architectures for Machine Perception.*, (Como, Italy), IEEE, 1995.

[2] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[3] S. Uras, F. Girosi, A. Verri, and V. Torre, “A computational approach to motion perception,” *Biological Cybernetics*, vol. 60, pp. 79–87, 1988.

[4] L. Wald, “Some terms of reference in data fusion,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, pp. 1190–1193, May 1999.

[5] R. C. Luo and M. G. Kay, “Multisensor integration and fusion in intelligent systems,” *IEEE Transactions on Systems Man and Cybernetics*, vol. 19, pp. 901–931, September/October 1989.

[6] B. V. Dasarathy, “Decision fusion strategies in multisensor environments,” *IEEE Transactions on Systems Man and Cybernetics*, vol. 21, pp. 1140–1154, September/October 1991.

[7] M. A. Abidi and R. C.Gonzales, eds., *Data Fusion in Robotics and Machine Intelligence*. Academic Press Inc., 1992.

[8] I. Bloch, “Information fusion operators for data fusion: A comparative review with classification,” *IEEE Transactions on Systems Man and Cybernetics*, vol. 26, pp. 52–67, January 1996.

[9] D. L. Hall and J. Llinas, “An introduction to multisensor data fusion,” *Proceedings of the IEEE*, vol. 85, pp. 6–23, January 1997.

[10] P. K. Varshney, “Multisensor data fusion,” *Electronics and Communication Engineering Journal*, vol. 9, pp. 245–253, December 1997.

[11] B. V. Dasarathy, “Elucidative fusion systems - an exposition,” *Information Fusion*, no. 1, pp. 5–15, 2000.

[12] T. Lee, J. A. Richards, and P. H. Swain, “Probablistic and evidential approaches for multisource data analysis,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 25, pp. 283–293, May 1987.

[13] S.Dellepiane, D.D.Giusto, S.B.Serpico, and G.Vernazza, “Information fusion by a knowledge-based system for SAR image interpretation,” in *IGARSS*, pp. 1845–1846, ESA, ESA Publications Divison, August 1988.

[14] D. Haverkamp and C. Tsatsoulis, “Information fusion for estimation of summer MIZ ice concentration from SAR imagery,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, pp. 1278–1291, March 1999.

[15] S. L. Hegárat-Mascle, I. Block, and D. Vidal-Madjar, "Application of Dempster-Shafer evidence theory to unsupervised classification in multisource remote sensing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, pp. 1018–1031, July 1997.

[16] V. Chatzis, A. G. Borş, and I. Pitas, "Multimodal decision-level fusion for person authentication," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 29, pp. 674–680, November 1999.

[17] J.Kittler, J.Matas, K.Jonsson, and M. Sanchez, "Combining evidence in personal identity verification systems," *Pattern Recognition Letters*, vol. 18, pp. 845–852, 1997.

[18] J. Kittler, Y. Li, J. Matas, and M. Sanchez, "Combining evidence in multimodal personal identity recognition systems.," in *International Conference on Audio- and Video-Based Biometric Person Authentification*, (Crans Montana, Switzerland), pp. 327–334, 1997.

[19] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz, "Fusion of face and speech data for person identity verification," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1065–1074, September 1999.

[20] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition.," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, pp. 418–419, May/June 1992.

[21] B. V. Dasarathy, "Intelligent learning techniques for multi-source information fusion environments," *Proceedings of the IEEE*, vol. 85, pp. 6–23, January 1997.

[22] Z. Chair and P. Varshney, "Optimal data fusion in multiple sensor detection systems," *IEEE Transaction on Aerospace and Electronic Systems*, vol. AES-22, pp. 98–101, January 1986.

[23] R. R. Tenney and J. Nils R. Sandell, "Detection with distributed sensors," *IEEE Transaction on Aerospace and Electronic Systems*, vol. AES-17, pp. 501–510, July 1981.

[24] L. A. Klein, *Millimeter-Wave and Infrared Multisensor Design and Signal Processing*. Artech House Inc., 1997.

[25] C. Kermad and K. Chehdi, "Multi-bands image segmentation: A scalar approach," in *IEEE International Conference on Image Processing*, vol. 3, (Vancouver, Canada), pp. 468–472, IEEE, September 2000.

[26] G.Anandalingam and L. Chen, "Linear combination of forcasts: a general bayesian model," *Journal of Forcasting*, vol. 8, pp. 199–214, 1989.

[27] A. Nejatali and I. R. Ciric, "Novel image fusion methodology using fuzzy set theory," *Optical Engineering*, pp. 485–491, February 1998.

[28] G.K.Matsopolous, S.Marshall, and N. J, "Multiresolution morphological fusion of MR and CT images of the human brain," *IEE Vision, Image and Signal Processing*, vol. 141, June 1994.

[29] F. Giorda and A. Racciu, "Bandwidth reduction of video signals via shift vector transmission," *IEEE Transactions on Communications*, vol. COMM-23, pp. 1002–1004, September 1975.

[30] A.N.Netravali and J.D.Robbins, "Motion-compensated television coding: Part 1," *The Bell Systems Technical Journal*, vol. 58, pp. 631–670, March 1979.

[31] I. J. 11172, "Coding of moving pictures and associated audio for storage at up to 1.5 mbit/s," tech. rep., ISO/IEC, November 1992.

[32] CCITT, "Recommendation H.261 video codec for audiovisual services at p*64 kbits/sec," Tech. Rep. COM XVR37-E, CCITT, August 1990.

[33] R. P. Wildes, "A measure of motion salience for surveillance applications," in *IEEE International Conference on Image Processing*, pp. 183–187, IEEE, 1998.

[34] F. Decroos, P. Schelkens, F. Stiens, J. Cornelis, and V. Christopoulos, "Motion monitoring and classification with centre-biased motion estimation," in *IEEE International Conference on Image Processing*, vol. 1, (Vancouver, Canada), pp. 872–875, IEEE, September 2000.

[35] A. Giachetti, M. Campani, and V. Torre, "The use of optical flow for road navigation," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 34–48, February 1998.

[36] N. P. Papanikololopolous, P. K. Khosla, and T. Kanade, "Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision," *IEEE Transactions on Robotics and Automation*, vol. 9, pp. 14–35, February 1993.

[37] C. E. Smith, C. A. Richards, S. A. Brandt, and N. P. Papanikololopolous, "Visual tracking for intelligent vehicle highway systems," *IEEE Transactions on Vehicular Technology*, vol. 45, pp. 744–759, November 1993.

[38] D. S. Kalivas and A. A. Sawchuk, "A region matching motion estimation algorithm," in *Computer Vision and Image Processing: Image Understanding*, vol. 54, pp. 275–288, September 1991.

[39] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, April 1991.

[40] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, IEEE, 1994.

[41] M. R. Luettgen, W. C. Karl, and A. S. Willsky, "Efficient multiscale regularization with applications to the computation of optical flow.," *IEEE Transactions on Image Processing*, vol. 3, pp. 41–63, January 1994.

[42] U.-V. Koc and K. J. R. Liu, "DCT pseudo-phase techniques for video coding and image registration," Tech. Rep. TSR TR 95-1, University of Maryland, 1994.

[43] U.-V. Koc and K. J. R. Liu, "Low-complexity motion estimation scheme utilizing sinusoidal orthogonal principle," in *IEEE Workshop on Visual Signal Processing*, (New Brunswick), pp. 57–62, IEEE, September 1994.

[44] U.-V. Koc and K. J. R. Liu, "Discrete-cosine/sine-transform based motion estimation," in *IEEE International Conference on Image Processing*, (Austin), pp. III–771–775, IEEE, November 1994.

[45] R. Young and N. Kingsbury, “Frequency-domain motion estimation using a complex lapped transform,” in *IEEE Transactions on Image Processing*, vol. 2, pp. 3–17, IEEE, 1993.

[46] 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.

[47] A.M.Peacock, D.Renshaw, and J.Hannah, “A fuzzy data fusion approach for image processing,” *IEE Electronics Letters*, vol. 35, pp. 1527–1529, September 1999.

[48] A.M.Peacock, D.Renshaw, and J.Hannah, “A fuzzy data fusion method for improved motion estimation,” in *Advanced Concepts in Intelligent Vision Systems*, (Baden-Baden, Germany), pp. 116–120, August 1999.

[49] A.M.Peacock, D.Renshaw, and J.M.Hannah, “Fusion of electrorotation frequency estimates.” accepted for publication in the International Journal of Information Fusion, 2001.

[50] A.M.Peacock, P.J.Edwards, D.Renshaw, and J.Hannah, “The effect of confidence errors on estimation fusion with bayes criterion,” in *Proceedings of Workshop on Intelligent Sensor Processing*, (Birmingham, UK), pp. 7/1–7/5, DERA/IEE, February 2001. ISSN 0963-3308.

[51] A.M.Peacock, D.Renshaw, J.Hannah, and P.Grant, “A data fusion method for improved motion estimation,” in *Proceedings of European Signal Processing Conference*, (Tampere, Finland), pp. 1485–1488, September 2000.

[52] A.M.Peacock, D.Renshaw, and J.Hannah, “A data fusion solution to the accuracy-efficiency trade-off problem in motion estimation,” in *Proceedings of IEEE International Conference on Image Processing*, vol. 1, (Vancouver, Canada), pp. 840–843, IEEE, September 2000.

[53] A.M.Peacock, D.Renshaw, and J.Hannah, “Motion direction estimates from differenced DCT images,” *IEE Electronics Letters*, vol. 37, pp. 163–164, February 2001.

[54] A.M.Peacock, S.Matsunaga, D.Renshaw, J.Hannah, and A.Murray, “Reference block updating when tracking with the block matching algorithm,” *IEE Electronics Letters*, vol. 36, pp. 309–310, February 2000.

[55] B. V. Dasarathy, “Intelligent learning techniques for multi-source information fusion environments,” in *IEEE Conference on Decision and Control*, (Tampa, Florida), pp. 221–226, IEEE, December 1998.

[56] A. Tirumalai, B.G.Schunck, and R.C.Jain, “Dynamic stereo with self-calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 1184–1189, December 1992.

[57] J.Shieh, H. Zhuang, and R. Sudhakar, “A direct method of motion estimation from a sequence of stereo images,” in *International Conference on Systems, Man and Cybernetics*, pp. 1229 –1234, IEEE, 1992.

[58] J.E.Higgins, R.I.Damper, and C.J.Harris, "A multi-spectral data fusion approach to speaker recognition," in *International Conference on Information Fusion*, (Sunnyvale, CA), pp. 1136–1143, 1999.

[59] H. L. V. Trees, *Detection, Estimation and Modulation Theory*, vol. 1. John Wiley and Sons, 1968.

[60] J.B.Allen, "How do humans recognise speech," *IEEE Transactions on Speech and Audio Processing*, vol. 2, pp. 567–577, October 1994.

[61] G. Shafer, *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[62] G.D.Jones, A.A.Hodgetts, R.E.Allsop, N.Sumpter, and M.-A. Vicencio-Silva, "A novel approach for surveilance using visual and thermal images," in *Proceedings of DERA/IEE Workshop on Intelligent Sensor Processing*, (Birmingham, UK), pp. 9/1–9/6, IEE, 2001.

[63] A. Toet, L. Ruyven, and J.M.Valeton, "Merging thermal and visual images by a contrast pyramid," *Optical Engineering*, pp. 789–792, July 1989.

[64] Z. Zhang and R. S. Blum, "A categorization and study of multiscale-decomposition-based image fusion schemes," *Proceedings of the IEEE*, pp. 1315–1328., Aug 1999.

[65] C.S.Xydeas and V.Petrovic, "Objective image fusion performance measure," *IEE Electronics Letters*, vol. 36, February 2000.

[66] L. G. Brown, "A survey of image registration techniques," *ACM Computer Surveys*, vol. 24, pp. 325–376, December 1992.

[67] H. H. Li, B.S.Manjunath, and S. K. Mitra, "A contour-based approach to multisensor image registration," *IEEE Transactions on Image Processing*, vol. 4, pp. 320–334, March 1995.

[68] H. H. Li and Y.-T. Zhou, "Automatic visual/IR image registration," *Optical Engineering*, vol. 35, pp. 391–400, February 1996.

[69] A. A. III, "Pyramidal techniques for multisensor fusion," *Proceedings SPIE*, vol. 1828, pp. 124–131, 1992.

[70] L. J. Chipman, T. M. Orr, and L. N. Graham, "Wavelets and image fusion," *Proceedings SPIE*, vol. 2569, pp. 208–219, 1995.

[71] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, July 1989.

[72] R. M. Rao and A. S. Bopardikar, *Wavelet Transforms*. Addison Wesley Longman, 1998.

[73] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, pp. 205–220, April 1992.

[74] C. M. Bishop, *Neural Networks for Pattern Recognition*, ch. 9. Clarendon Press, Oxford, 1995.

[75] P. Edwards and A. Murray, "Committee formation for reliable and accurate neural prediction in industry," in *Proceedings of the European Symposium on Artificial Neural Networks*, (Bruges, Belgium), pp. 141–146, April 2000.

[76] G. Hinton, "Products of experts," in *Proceedings of the Ninth International Conference on Artificial Neural Networks*, vol. 1, pp. 1–6, 1999.

[77] R. A. Jacobs and M. I. Jordan, "A competitive modular connectionist architecture," in *Neural Information Processing Systems*, vol. 3, pp. 767–773, Morgan Kaufmann Publishers, 1991.

[78] S. J. Nowlan and G. E. Hinton, "Evaluation of adaptive mixtures of competing experts," in *Neural Information Processing Systems*, vol. 3, pp. 774–780, Morgan Kaufmann Publishers, 1991.

[79] D.E.Rumelhart and D.Zipser, "Feature discovery by competitive learning," *Cognitive Science*, vol. 9, 1985.

[80] D. K. McNeill and H. C. Card, "Competitive learning for extraction of visual representations of motion," in *IEEE International Joint Conference on Neural Networks*, (Washington, D. C.), 1999.

[81] S. C. Thomopoulos, R. V. Viswanathan, and D. C. Bougoulias, "Optimal decision fusion in multiple sensor systems," *IEEE Transaction on Aerospace and Electronic Systems*, vol. AES-23, pp. 644–653, September 1987.

[82] V. Aalo and R. Viswanathan, "Multilevel quantisation and fusion scheme for the decentralised detection of an unknown signal," *IEE Proceedings of Radar, Sonar Navigation*, vol. 141, pp. 37–44, February 1994.

[83] N. Ansari, J.-G. Chen, and Y.-Z. Zhang, "Adaptive decision fusion for unequiprobable sources," *IEE Proceedings of Radar, Sonar Navigation*, vol. 144, pp. 105–111, June 1997.

[84] J. Kittler, M. Hatlef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 226–239, March 1998.

[85] J. Kittler, A. Hojjatolesami, and T. Windeatt, "Weighting factors in multiple expert fusion," in *British Machine Vision Conference*, (Colchester, England), pp. 41–50, 1997.

[86] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 66–75, January 1994.

[87] V.D.Mazurov, A.I.Krivonogov, and V.S.Kazantsev, "Solving of optimization and identification problems by the committee methods," *Pattern Recognition*, vol. 4, no. 20, pp. 371–378, 1987.

[88] Y.Yussoff, J.Kittler, and W.Christmas, "Combining multiple experts for classifying shot changes in video sequences," *6th International Conference on Multimedia Computing and Systems*, June 1999.

[89] L. Hong and G.J.Wang, “Centralised integration of multisensor noisy and fuzzy data,” *IEE Proceedings of Control Theory and Applications*, vol. 142, pp. 459–465, September 1995.

[90] D. Dubois and H. Prade, *Fuzzy Sets and Systems*, vol. 144 of *Mathematics in Science and Engineering.* Academic Press, 1980.

[91] J.D.Jobson, *Applied Multivariate Data Analysis*, vol. 2:Categorical and Multivariate Methods. Springer-Verlag, 1992.

[92] G.Stockmann, “Object recognition and locatization via pose clustering,” *Computer Vision, Graphics and Image Processing*, vol. 40, pp. 361–387, 1987.

[93] W.J.Austin and A.M.Wallace, “Object location by parallel pose clustering,” *Computer Vision and Image Understanding*, vol. 72, no. 3, pp. 304–327, 1998.

[94] D.H.Ballard, “Generalising the hough transform to detect arbitrary shapes,” *Pattern Recognition*, vol. 13, pp. 111–122, 1981.

[95] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, pp. 338–353, 1965.

[96] H. J. Zimmermann, *Fuzzy Set Theory and its Applications*. Klewer Academic Publishers, 1991.

[97] S. Fruhwirth-Schnatter, “On statistical inference for fuzzy data with applocations to descriptive statistics,” *Fuzzy Sets and Systems*, vol. 50, pp. 143–165, 1992.

[98] S. Bozic, *Digital and Kalman Filtering.* Edward Arnold, 1979.

[99] M. D. Out and W. A. Kosters, “A bayesian approach to combined neural networks forcasting,” in *Proceedings of the European Symposium on Artificial Neural Networks*, (Bruges, Belgium), pp. 323–328, 2000.

[100] J. Kittler, M. Hatlef, and R. P. Duin, “Combining classifiers,” in *International Conference on Pattern Recognition*, pp. 897–901, IEEE, 1996.

[101] M. Laviolette and J. W. S. Jr, “The efficacy of fuzzy representations of uncertainty,” *IEEE Transactions on Fuzzy Systems*, vol. 2, pp. 4–15, February 1994.

[102] B. V. Dasarathy, “Intelligent multi-classifier fusion for decision making in ballistic missile defense applications,” in *IEEE Conference on Decision and Control*, (Tampa, Florida), pp. 233–238, IEEE, December 1998.

[103] M. Isard and A. Blake, “Condensation - conditional density propagation for visual tracking,” *International Journal Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[104] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, “Bayesian model averaging: A tutorial,” *Statistical Science*, vol. 14, no. 4, pp. 382–417, 1999.

[105] A. Mitiche and P. Bouthemy, “Computation and analysis of image motion: A synopsis of current problems,” *International Journal of Computer Vision*, vol. 19, pp. 29–55, 1996.

[106] S. Beauchemin and J. Barron, “The computation of optical flow,” *ACM Computing Surveys*, vol. 27, 1995.

[107] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt, “Performance of optical flow techniques,” Tech. Rep. 229, Department of Computer Science, University of Western Ontario, London, Ontario, Canada N6A 5B7, July 1992.

[108] J. Malo, F. Ferri, J. Albert, and J. Artigas, “Splitting criterion for hierarchical motion estimation based on perceptual coding,” *IEE Electronics Letters*, vol. 34, pp. 541–543, March 1998.

[109] V. Seferidis and M. Ghanbari, “General approach to block-matching motion estimation,” *Optical Engineering*, vol. 32, pp. 1464–1474, July 1993.

[110] V. Seferidis and M. Ghanbari, “Generalised block-matching motion estimation using quad-tree structured spatial decomposition,” *IEE Vision, Image and Signal Processing*, vol. 141, pp. 446–452, December 1994.

[111] T.Kanade and M.Okutomi, “A stereo matching algorithm with an adaptive window: theory and experiment,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, pp. 920–932, September 1994.

[112] A. Fusiello, V. Roberto, and E. Trucco, “Symmetric stereo with multiple windowing,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 8, pp. 1053–1066, 2000.

[113] J.Li and X.Lin, “Sequential image coding based on multiresolution tree architecture,” *IEE Electronics Letters*, vol. 29, pp. 1545–1547, August 1993.

[114] W. Li and E. Salari, “Successive elimination algorithm for motion estimation,” in *Image Processing*, vol. 4, pp. 105–107, January 1995.

[115] C.-H. Lee and L.-H. Chen, “A fast motion estimation algorithm based on the block sum pyramid,” *IEEE Transactions on Image Processing*, vol. 6, pp. 1587–1591, November 1997.

[116] J. Jain and A. Jain, “Displacement measurement and its application in interframe image coding,” *IEEE Transactions on Communications*, vol. COM-29, pp. 1799–1808, December 1981.

[117] Y. Ninomiya and Y. Ohtsuka, “A motion-compensated interframe coding scheme for television pictures,” *IEEE Transactions on Communications*, vol. 30, pp. 201–211, January 1982.

[118] T. Koga, K.Iinuma, A.Hirano, Y.Iijima, and T.Ishiguro, “Motion compensated interframe coding for video conferencing,” in *National Telecommunications Conference*, (New Orleans, LA), pp. G5.3.1–5.3.5, November 1981.

[119] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 438–442, August 1994.

[120] L.-M. Po and W.-C. Ma, “A novel four-step search algorithm for fast block motion estimation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 313–317, June 1996.

[121] H. Nisar and T.-S. Choi, “An advanced centre biased search algorithm for motion estimation,” in *IEEE International Conference on Image Processing*, vol. 1, (Vancouver, Canada), pp. 832–835, IEEE, September 2000.

[122] M. Chen, L. Chen, K. Cheng, and M. Chen, “Efficient hybrid tree/linear array architectures for block-matching motion estimation algorithms,” *IEE Vision, Image and Signal Processing*, vol. 143, pp. 217–222, August 1996.

[123] J. Feng, K. Lo, H. Mehrpour, and A. Karbowiak, “Adaptive block matching algorithm for video compression,” *IEE Vision, Image and Signal Processing,*, vol. 145, pp. 173–178, June 1998.

[124] M. Z. Coban and R. M. Mersereau, “A fast exhaustive search algorithm for rate constrained motion estimation,” *IEEE Transactions on Image Processing*, vol. 7, pp. 769–773, May 1998.

[125] J. Fan and F. Gan, “Motion estimation based on global and local uncompensablility analysis,” *IEEE Transactions on Image Processing*, vol. 6, pp. 1584–1587, November 1997.

[126] J.O.Limb and J.A.Murphy, “Measuring the speed of moving objects from television signals.,” *IEEE Transactions on Communications*, vol. 23, pp. 474–478, April 1975.

[127] J.O.Limb and J.A.Murphy, “Estimating the velocity of moving images in television signals.,” *Computer Graphics and Image Processing*, vol. 4, pp. 311–327, February 1975.

[128] C. Cafforio and F. Rocca, “Methods for measuring small displacements of television images,” *IEEE Transactions on Information Theory*, vol. IT-22, September 1976.

[129] U.-V. Koc and K. J. Liu, “DCT based motion estimation,” *IEEE Transactions on Image Processing*, vol. 7, pp. 948–965, July 1998.

[130] R. C. Gonzales and R. E. Woods, *Digital Image Processing*. Addison-Wesley Publishing Company, 1993.

[131] P. Yip and K. R. Rao, “On the shift property of DCT’s and DST’s,” in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, pp. 404–406, IEEE, March 1987.

[132] D. S. Kim and S. U. Lee, “Image vector quantizer based on classification in the DCT domain,” *IEEE Transactions on Communications*, vol. 39, pp. 549–556, April 1991.

[133] G. C. M. H. Lee, “Classified vector quantisation with variable block-size DCT models,” *IEE Proceedings Vision and Image Signal Processing*, vol. 141, pp. 39–48, February 1994.

[134] T. J. D. C. L. Pagliari, “Disparity estimation using edge-oriented classification in the DCT domain,” *IEE Electronics Letters*, vol. 34, pp. 1214–1216, June 1998.

[135] "Hamburg taxi sequence." http://www.cs.ubc.ca/nest/lci/vista/index-pix.html. Originally captured at the University of Hamburg.

[136] H. Liu, T.-H. Hong, M. Herman, and R. Chellappa, "Accuracy vs. efficiency trade-offs in optical flow algorithms," *Computer Vision and Image Understanding*, vol. 72, pp. 271–286, December 1998.

[137] H.Seibert and W.Engelhardt, "Performance data and limiting conditions of object tracking algorithms in the presence of camera noise," *IEE Proceedings*, vol. 135, pp. 111–113, February 1988.

[138] D. N.Bhat and S. K.Nayar, "Ordinal methods for image correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 415–423, April 1998.

[139] B. Klaus and P. Horn, *Robot Vision*. The MIT Press, 1986.

[140] D. Ballard and C. Brown, *Computer Vision*. Prentice-Hall, 1982.

[141] F.Glazer, G.Reynolds, and P.Anandan, "Scene matching by hierarchical correlation," in *IEEE Conference on Computer Vision and Pattern Recognition*, (Annapolis), pp. 432–441, IEEE, 1983.

[142] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, pp. 283–310, 1989.

[143] A.Bainbridge-Smith and R.G.Lane, "Measuring confidence in optical flow estimation," *IEE Electronics Letters*, vol. 32, pp. 882–884, May 1996.

[144] M.Fleury, A.C.Downton, and A.F.Clark, "Confidence testing optical flow estimates," *IEE Electronics Letters*, vol. 34, pp. 446–447, March 1998.

[145] T. H. Corben, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. McGraw-Hill, (16th printing) 1996. ISBN 0-07-013143-0.

[146] H. Pohl, *Dielectrophoresis*. Cambridge University Press, UK, 1978.

[147] R. P. X-F. Zhou, J. P. H. Burt, "Automatic cell electrorotation measurements: Studies of the biological effects of low-frequency magnetic fields and of heat shock," *J. Phys. Med. Biol*, vol. 43, pp. 1075–1090, 1998.

[148] C. Dalton, R. Pethig, C. A. Paton, and H. V. Smith, "Electrorotation of oocysts of cyclospora cayetanensis," in *Institute of Physics Conference Series*, vol. 163, pp. 85–88, March 1999.

[149] R. Pethig and G. H. Markx, "Applications of dielectrophoresis in biotechnology," *Trends in Biotechnology*, vol. 15, pp. 426–432, 1997.

[150] J. P. H. B. A. Parton, R. Pethig, "Methods of analysis." Patent Application GB92/02705.

[151] S. Smith and J. Brady, "SUSAN - a new approach to low level image processing," *International Journal of Computer Vision*, vol. 23, pp. 45–78, 1997.

[152] E. V. R. D. Bella, A. B. Barclay, R. L. Eisner, and R. W. Schafer, “A comparison of rotation-based methods for iterative reconstruction algorithms,” *IEEE Transactions on Nuclear Science*, vol. 43, pp. 3370–3376, December 1996.

[153] G. James, *Advanced Engineering Mathematics*. Wokingham, England: Addison Wesley, 1994.

[154] P. Huber, D. Burley, D. Clements, P. Dyke, J. Searl, and J. Wright, *Robust statistics*. New York: Wiley, 1981.

[155] F. R. Hampel, E. M.Ronchetti, P. J.Rousseeuw, and W. A. Stahel, *Robust statistics, The approach based on influence functions.* New York: Wiley, 1986.

[156] E.Brookner, *Tracking and Kalman Filtering Made Easy*. John Wiley and Sons publ., 1998.

[157] A. Singh, “Incremental estimation of image-flow using a kalman filter,” in *IEEE Workshop on Visual Motion*, (Princeton, NJ), pp. 36–43, IEEE, 1991.

[158] Message Passing Interface Forum, “MPI: A message passing interface standard (v1.1),” tech. rep., University of Tennessee, June 1995.

[159] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, “A high-performance, portable implementation of the MPI message passing interface standard,” *Parallel Computing*, vol. 22, pp. 789–828, Sept. 1996.

[160] R. Alasdair, A. Bruce, J. G. Mills, and A. G. Smith, “CHIMP/MPI user guide,” Tech. Rep. EPCC-UG1994-05, Edinburgh Parallel Computing Centre (EPCC), Edinburgh, UK, April 1994.

[161] C.D.Haworth, A.M.Peacock, and D.Renshaw, “Performance of reference block updating techniques when tracking with the block matching algorithm.” to appear in Proceedings of IEEE International Conference on Image Processing, 2001.

[162] R.E.Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME-Journal of Basic Engineering*, pp. 35–45, March 1960.

[163] S. Oualline, *Practical C++ Programming*. O’Reilly and Associates, Inc., 1997. ISBN 1-56592-139-9.

Appendix A
Publications

A.1 Refereed Journals

A.M.Peacock, D.Renshaw and J.M.Hannah, *Motion direction estimates from differenced DCT images* in IEE Electronics Letters, Vol 37(03), pp 163-164, February 2001.

A.M.Peacock, D.Renshaw, J.M.Hannah, *Fusion of Electrorotation Frequency Estimates*, To appear in the International Journal of Information Fusion, 2001.

A.M.Peacock, S.Matsunaga, D.Renshaw, J.Hannah and A.Murray, *Reference Block Updating when Tracking with the Block Matching Algorithm* in IEE Electronics Letters, Vol 36(04), pp 309-310, February 2000.

A.M.Peacock, D.Renshaw and J.Hannah, *A Fuzzy Data Fusion Approach for Image Processing*, in IEE Electronics Letters, Vol 35(18), pp 1527-1529, September 1999.

A.2 Refereed Conferences

C.D.Haworth, A.M.Peacock, D.Renshaw, *Performance of Reference Block Updating Techniques when Tracking with the Block Matching Algorithm*, to appear in Proceedings of IEEE ICIP (International Conference on Image Processing), 2001, Greece.

A.M.Peacock, D.Renshaw, J.Hannah, *A Data Fusion Solution to the Accuracy-Efficiency Trade-off Problem in Motion Estimation*, Vol 1, pp 840-843, Proceedings of IEEE ICIP (International Conference on Image Processing), September 2000, Vancouver, Canada. ISBN 0-7803-6300-0

A.M.Peacock, D.Renshaw and J.Hannah, *A Data Fusion Method for Improved Motion Estimation*, Proceedings of EUSIPCO (European Signal Processing Conference), pp 1485-1488, September 2000, Tampere, Finland. ISBN 952-15-0443-9

A.M.Peacock, D.Renshaw and J.Hannah, *A Fuzzy Data Fusion Method for Improved Motion*

Estimation, in Proceedings of ACIVS, August 1999, Baden-Baden, Germany. ISBN 0-921836-74-0

A.3 Workshops

A.M.Peacock, P.J.Edwards, D.Renshaw and J.Hannah, *The Effect of Confidence Errors on Estimation Fusion Using Bayes' Criterion*, to appear in proceedings of DERA/IEE Workshop on Intelligent Sensor Processing, 14th February 2001, Birmingham UK.

Appendix B
Background Derivations / Algorithms

B.1 Derivation of Horn and Schunck's Iterative Solution

The purpose of this section is to set down all the steps of the derivation of the optical flow method proposed by Horn and Schunck in [2]. What is presented here is the discrete derivation discussed by Horn and Klaus in [139], with the approximations and all derivation steps made explicit.

In their method, Horn and Schunck assume that the intensity of an imaged point remains constant over time and motion. Let $I(x, y, t)$ be the intensity of pixel (x, y) at time t, then:

$$\frac{dI}{dt} = 0 \tag{B.1}$$

which can be expanded using the Chain Rule for differentiation to give:

$$\Rightarrow \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{B.2}$$

where $\frac{dx}{dt}$ and $\frac{dy}{dt}$ are the horizontal u and vertical v components of the velocity respectively. This is the *Motion Gradient* constraint E_M. Setting $\frac{\partial I}{\partial x} = I_x$, $\frac{\partial I}{\partial y} = I_y$, $\frac{\partial I}{\partial t} = I_t$, (B.2) can be written as:

$$E_M = I_x u + I_y v + I_t = 0 \tag{B.3}$$

The spatial and temporal derivatives I_x, I_y and I_t are approximated by the differences in a cube of pel intensity values, as illustrated in Figure(B.1) [2, 139] and given below:

$$I_x = \begin{bmatrix} I(i+1,j,k) & -I(i,j,k)+ \\ I(i+1,j,k+1) & -I(i,j,k+1)+ \\ I(i+1,j+1,k) & -I(i,j+1,k)+ \\ I(i+1,j+1,k+1) & -I(i,j+1,k+1) \end{bmatrix} /4 \tag{B.4}$$

$$I_y = \begin{bmatrix} I(i,j+1,k) & -I(i,j,k)+ \\ I(i,j+1,k+1) & -I(i,j,k+1)+ \\ I(i+1,j+1,k) & -I(i+1,j,k)+ \\ I(i+1,j+1,k+1) & -I(i+1,j,k+1) \end{bmatrix} /4 \tag{B.5}$$

$$I_z = \begin{bmatrix} I(i,j,k+1) & -I(i,j,k)+ \\ I(i,j+1,k+1) & -I(i,j+1,k)+ \\ I(i+1,j,k+1) & -I(i+1,j,k)+ \\ I(i+1,j+1,k+1) & -I(i+1,j+1,k) \end{bmatrix} /4 \tag{B.6}$$

(B.3) is the equation of a line in velocity space. The motion estimate is constrained to lie on this line. To constrain the estimate to a single point, regularisation constraints are required. See [41] for an overview of regularisation techniques. Horn and Schunck suggest the *smoothness constraint* E_S , which requires the motion vectors to vary smoothly over neighbouring pixels:

$$E_S = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{B.7}$$

A weighted combination of the smoothness and motion gradient constraints gives an error E to minimise over all the pel locations (i,j) in the image:

$$E = \sum_i \sum_j \left(E_m^2 + \lambda E_s\right) \tag{B.8}$$

where λ is the weighting term (often expressed as $\alpha^2 = \lambda$ in the literature). To minimise this equation for a motion vector at pel location (k,l), derivatives are taken with respect to u_{kl} and

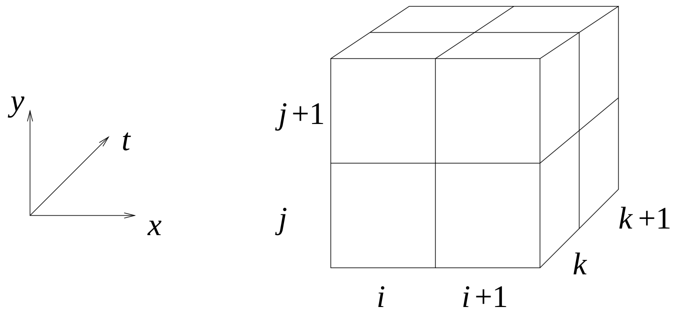


Figure B.1: *Cube used for approximating spatial and temporal derivatives with 4 point intensity differences.*

-1/12	-1/6	-1/12
-1/6	1	-1/6
-1/12	-1/6	-1/12

Figure B.2: *Convolution kernel used to approximate the Laplacian of the pel intensities.*

v_{kl}, and the result set to 0:

$$\frac{\partial E}{\partial u_{kl}} = 2\left(I_{x_{kl}}u_{kl} + I_{y_{kl}}v_{kl} + I_{tkl}\right)I_{x_{kl}} + 2\lambda\left(\frac{\partial^2 u_{kl}}{\partial x^2} + \frac{\partial^2 u_{kl}}{\partial y^2}\right) = 0 \qquad \text{(B.9)}$$

$$\frac{\partial E}{\partial v_{kl}} = 2\left(I_{x_{kl}}u_{kl} + I_{y_{kl}}v_{kl} + I_{tkl}\right)I_{y_{kl}} + 2\lambda\left(\frac{\partial^2 v_{kl}}{\partial x^2} + \frac{\partial^2 v_{kl}}{\partial y^2}\right) = 0 \qquad \text{(B.10)}$$

The second term on the RHS is the Laplacian of the pel intensities, which Horn and Schunck [2] approximate by (see also Figure B.2):

$$\frac{\partial^2 u_{kl}}{\partial x^2} + \frac{\partial^2 u_{kl}}{\partial y^2} = u_{kl} - \bar{u}_{kl} \qquad \text{(B.11)}$$

where $\bar{u}_{kl}$ is defined as:

$$\begin{aligned}\bar{u}_{kl} = \frac{1}{12} & \big[&& I(i-1,j-1,t) + I(i+1,j-1,t) + \\ & && I(i-1,j+1,t) + I(i+1,j+1,t)\big] + \\ \frac{1}{6} & \big[&& I(i,j-1,t) + I(i,j+1,t) + \\ & && I(i-1,j,t) + I(i+1,j,t)\big] \end{aligned} \tag{B.12}$$

and so from (B.9) and (B.10):

$$\frac{\partial E}{\partial u_{kl}} \approx 2\left(I_{x_{kl}} u_{kl} + I_{y_{kl}} v_{kl} + I_{t_{kl}}\right) I_{x_{kl}} + 2\lambda(u_{kl} - \bar{u}_{kl}) \tag{B.13}$$

$$\frac{\partial E}{\partial v_{kl}} \approx 2\left(I_{x_{kl}} u_{kl} + I_{y_{kl}} v_{kl} + I_{t_{kl}}\right) I_{y_{kl}} + 2\lambda(v_{kl} - \bar{v}_{kl}) \tag{B.14}$$

Where (B.8) is at a minimum, the derivatives must be 0, so:

$$(I_{x_{kl}}^2 + \lambda) u_{kl} + I_{x_{kl}} I_{y_{kl}} v_{kl} = \lambda \bar{u}_{kl} - I_{x_{kl}} I_{t_{kl}} \tag{B.15}$$

$$I_{x_{kl}} I_{y_{kl}} u_{kl} + (I_{y_{kl}}^2 + \lambda) v_{kl} = \lambda \bar{v}_{kl} - I_{y_{kl}} I_{t_{kl}} \tag{B.16}$$

This can be rewritten in matrix form as follows:

$$\begin{bmatrix} \lambda I_{x_{kl}}^2 + \lambda & I_{x_{kl}} I_{y_{kl}} \\ I_{x_{kl}} I_{y_{kl}} & I_{y_{kl}}^2 + \lambda \end{bmatrix} \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix} = \begin{bmatrix} \lambda \bar{u}_{kl} - I_{x_{kl}} I_{t_{kl}} \\ \lambda \bar{v}_{kl} - I_{y_{kl}} I_{t_{kl}} \end{bmatrix} \tag{B.17}$$

In order to get this in terms of u_{kl} and v_{kl}, the inverse of the 2×2 matrix on the LHS must be found. The determinant of the matrix is:

$$\lambda^2 + \lambda I_{x_{kl}}^2 + \lambda I_{y_{kl}}^2 \tag{B.18}$$

and so (B.17) can be written:

$$(\lambda^2 + \lambda I_{x_{kl}}^2 + \lambda I_{y_{kl}}^2) \begin{bmatrix} u_{kl} \\ v_{kl} \end{bmatrix} = \begin{bmatrix} I_{y_{kl}}^2 + \lambda & -\lambda I_{x_{kl}} I_{y_{kl}} \\ -\lambda I_{x_{kl}} I_{y_{kl}} & I_{x_{kl}}^2 + \lambda \end{bmatrix} \begin{bmatrix} \lambda \bar{u}_{kl} - I_{x_{kl}} I_{t_{kl}} \\ \lambda \bar{v}_{kl} - I_{y_{kl}} I_{t_{kl}} \end{bmatrix} \tag{B.19}$$

This gives 2 equations, one for u and one for v:

$$u_{kl} = \frac{(I_{y_{kl}}^2 + \lambda)\bar{u}_{kl} - I_{x_{kl}} I_{y_{kl}} \bar{v}_{kl} - I_{x_{kl}} I_{t_{kl}}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.20}$$

$$v_{kl} = \frac{(I_{x_{kl}}^2 + \lambda)\bar{v}_{kl} - I_{x_{kl}} I_{y_{kl}} \bar{u}_{kl} - I_{y_{kl}} I_{t_{kl}}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.21}$$

The Gauss-Seidel method can then be used to give an iterative solution:

$$u_{kl}^{n+1} = \frac{(I_{y_{kl}}^2 + \lambda)\bar{u}_{kl}^n - I_{x_{kl}} I_{y_{kl}} \bar{v}_{kl}^n - I_{x_{kl}} I_{t_{kl}}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.22}$$

$$v_{kl}^{n+1} = \frac{(I_{x_{kl}}^2 + \lambda)\bar{v}_{kl}^n - I_{x_{kl}} I_{y_{kl}} \bar{u}_{kl}^n - I_{y_{kl}} I_{t_{kl}}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.23}$$

Noting that:

$$\frac{(I_{y_{kl}}^2 + \lambda)\bar{u}_{kl}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} = \bar{u}_{kl} - \frac{I_{x_{kl}}^2 \bar{u}_{kl}}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.24}$$

and similary for the vertical component, equations (B.22) and (B.23) are usually rearranged as follows [2, 107, 139]:

$$u_{kl}^{n+1} = \bar{u}_{kl}^{n} - \frac{I_{x_{kl}}(I_x\bar{u}_{kl}^{n} + I_{y_{kl}}\bar{v}_{kl}^{n} + I_{t_{kl}})}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.25}$$

$$v_{kl}^{n+1} = \bar{v}_{kl}^{n} - \frac{I_{y_{kl}}(I_x\bar{u}_{kl}^{n} + I_{y_{kl}}\bar{v}_{kl}^{n} + I_{t_{kl}})}{\lambda + I_{x_{kl}}^2 + I_{y_{kl}}^2} \tag{B.26}$$

B.2 The Kalman Filter

The Kalman filter [98, 156], originally proposed by R.E.Kalman in 1960 [162], is a popular filter that is referred to in a number of places in this thesis. This section gives a brief overview of the filter equations.

Filtering is used to maintain estimates $\hat{\mathbf{x}}_n$ of a state vector $\mathbf{x}$ at time n. The Kalman filter assumes a model of the observed process, this is formulated as follows:

$$\mathbf{x}_n = \Phi\mathbf{x}_{n-1} + \mathbf{q}_{n-1} \tag{B.27}$$

where Φ is the state transition matrix. $\mathbf{q}_{n-1}$ is an input to the process which is often used to model noise and so is called the *dynamic model noise*. Observations $\mathbf{y}_n$ of the state vector at time n are modelled by the true state plus some measurement error $\mathbf{u}_n$:

$$\mathbf{y}_n = H_n\mathbf{x}_n + \mathbf{u}_n \tag{B.28}$$

where H_n models the conversion from state space to observation space.

There are two sets of equations for the Kalman filter: predictor equations which predict the state vector $\mathbf{x}_n$ and error covariance matrix S_n, and corrector equations which correct the predictions given a new observation. This is illustrated in Figure B.3. The predictor equations are as follows, estimated values as opposed to true values are indicted by a ˆ:

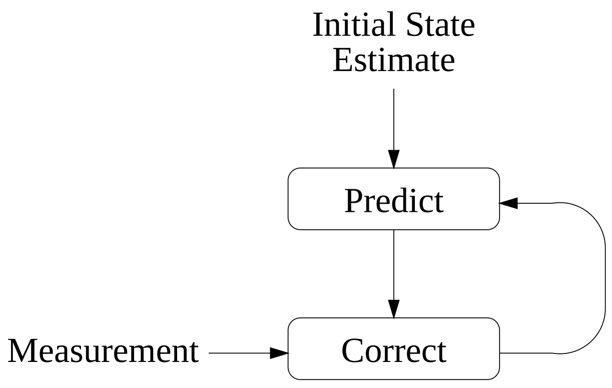


Figure B.3: *The Kalman filter comprises a prediction stage that predicts the state and error covariance matrix and a correction stage that corrects the predictions given a new observation.*

$$\begin{aligned} \hat{\mathbf{x}}_{n,n-1} &= \Phi\hat{\mathbf{x}}_{n-1} && \text{(B.29)} \\ \hat{S}_{n,n-1} &= \Phi\hat{S}_{n-1}\Phi^T + Q_n && \text{(B.30)} \end{aligned}$$

where Q_n is the covariance matrix of the dynamic model noise $\mathbf{q}_n$. $\hat{\mathbf{x}}_{n,n-1}$ is the predicted state estimate at time n, with the second term in the subscript indicating that this prediction is based on the state estimate at time $n-1$. Similarly, $\hat{S}_{n,n-1}$ is the predicted estimate error covariance given the estimated error covariance at time $n-1$.

The corrector equations are as follows:

$$\begin{aligned} \hat{\mathbf{x}}_n &= \hat{\mathbf{x}}_{n,n-1} + K_n(\mathbf{y}_n - H_n\hat{\mathbf{x}}_{n,n-1}) && \text{(B.31)} \\ \hat{S}_{n,n} &= \hat{S}_{n,n-1} - K_nH_n\hat{S}_{n,n-1} && \text{(B.32)} \end{aligned}$$

where K_n is the Kalman gain, calculated as follows:

$$K_n = \hat{S}_{n,n-1}H^T(R_n + H_n\hat{S}_{n,n-1}H^T)^{-1} \tag{B.33}$$

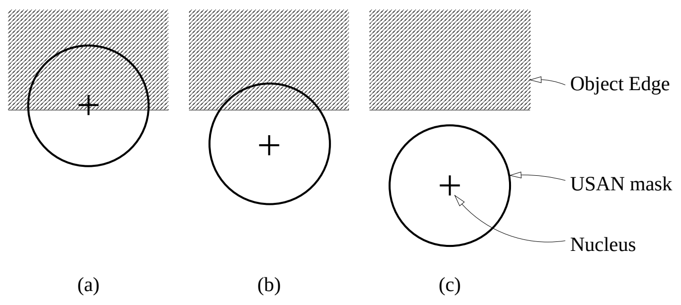


Figure B.4: *Example USAN masks, a) on an edge, b) near an object edge, c) on an area of constant intensity.*

where R_n is the covariance matrix of the measurement noise.

B.3 The SUSAN Edge Detector

The SUSAN approach to edge detection, proposed by Smith and Brady [151], was used in Chapter 4 to detect the edges of cells in dielectrophoresis image sequences. This approach uses a circular mask to identify features of interest in images. The intensity of the pels in the mask are compared to the intensity of the middle pel (the nucleus) to determine how much of the mask has a similar intensity to the nucleus. The mask is termed the Univalue Segment Assimilating Nucleus (USAN).

As illustrated in Figure B.4, where the mask is over an area of near constant intensity, most of the mask will have a similar intensity to the nucleus. Where the mask falls on an edge, only around half the mask will have the same intensity as the nucleus. Thus edges can be identified by finding where the USAN has a small value, this gives rise to the term Smallest USAN or SUSAN.

Figure B.5 shows the results of applying the SUSAN edge detector to an example frame from the Hambug Taxi sequence. As pels are discrete it is necessary to approximate the circular masks, two example mask sizes are shown. The intensity of pels in the result images is inversely proportional to the number of pels in the mask that similar to (within 5 grayscale intensity values of) the nucleus. High intensities in the result clearly correspond to edges in the original image.

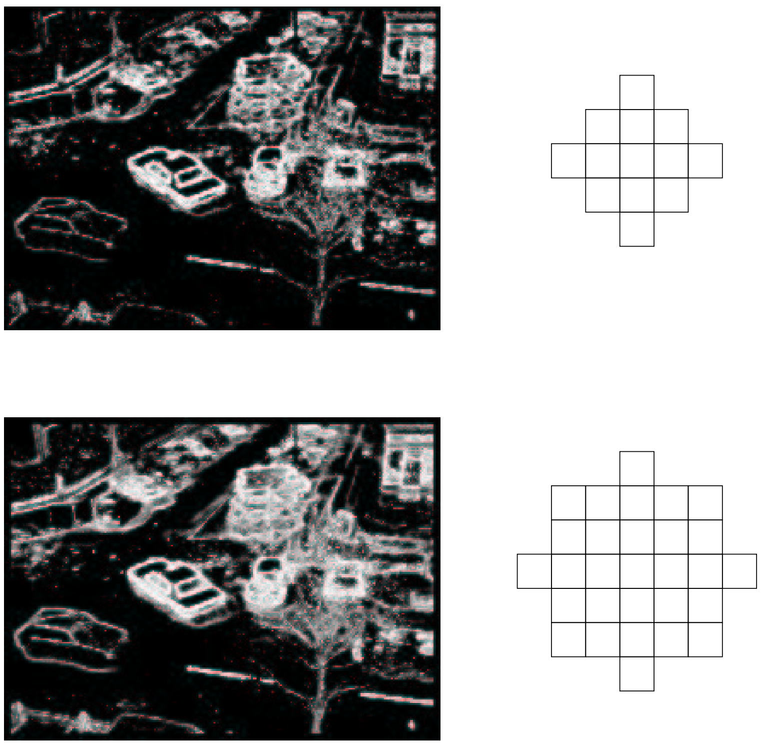


Figure B.5: *Results from applying the SUSAN edge detector to an example frame from the Hamburg Taxi sequence. Two examples of approximated circular masks are shown.*

Appendix C
Test Data

As well as using standard data sets, this thesis has introduced new data sets. This Appendix describes these data sets and explains why they were considered necessary. A list of the data sets with their dimensions and lengths is given in Table C.1.

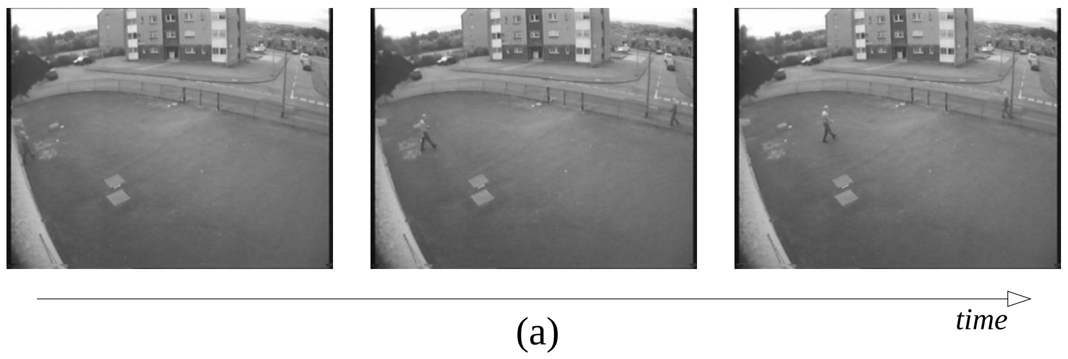
C.1 Car Park Sequences

The Car Park sequences are used in Chapter 4 to test reference block update strategies proposed by the author. These sequences were captured on a security camera over looking a church car park, and 2 hours of footage were stored on VHS video tape. As a result, the sequences are of low quality and so are difficult for tracking systems to cope with. The author thanks Dr J M Hannah for making this video tape available.

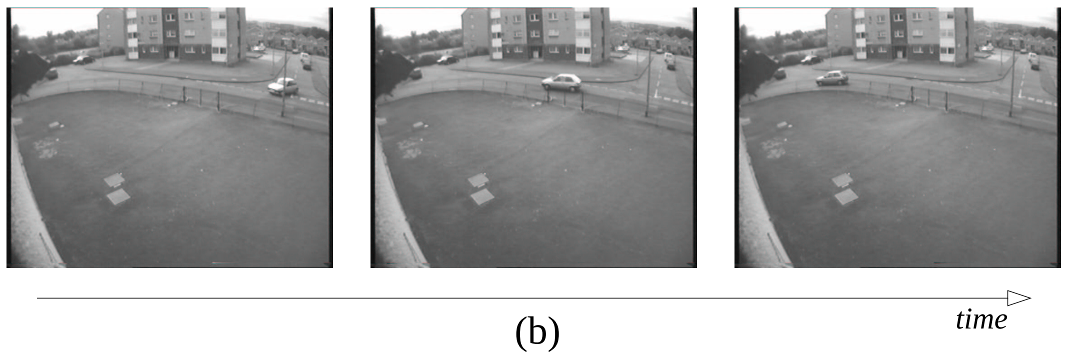
The author converted two sections of the VHS footage to grayscale PGM files using commercial image processing packages Adobe Premiere and Adobe PaintShopPro. The Car Park A sequence shows a pedestrian walking across the car park, the Car Park B sequence shows a car moving behind the railings on the car park border. Example frames from the Car Park A and Car Park B sequences are shown in Figure C.1.

Sequence	Image Size	Number of frames
Car Park A	720×576	75
Car Park B	720×576	106
Car Following	320×240	100
Advancing Photo	360×288	12
Translating Photo	280×200	41

Table C.1: *New experimental test image sequences used in this thesis.*



(a)



(b)

Figure C.1: *Example frames from the start, middle and end of a) the Car Park A sequence showing a pedestrian walking across the car park, b) the Car Park B sequence showing a car moving behind the railings on the car park border.*

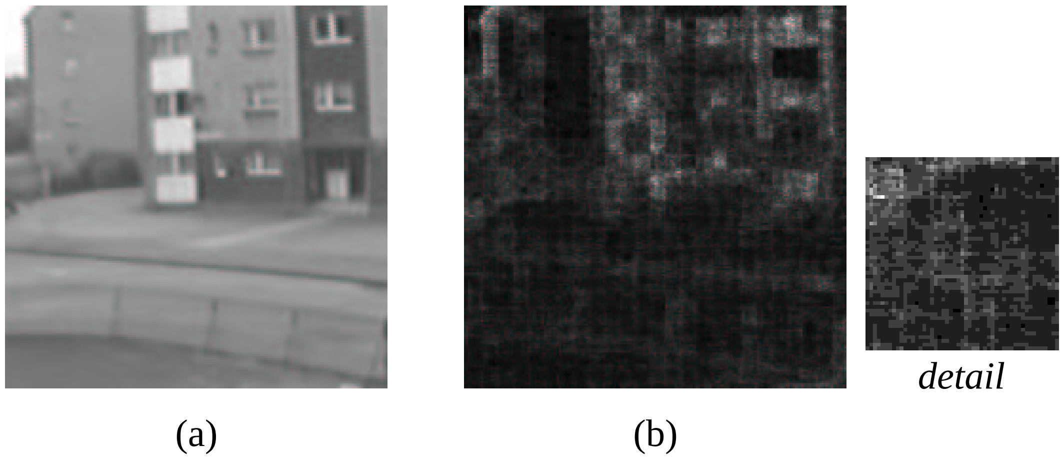


Figure C.2: *200×200 pel section of the Car Park A sequence. a) mean intensity values, b) variance in the intensity values, with detail 50×50 pel block from the centre of the segment, illustrating the blocking artifacts present in the data.*

As a result of the format conversions, 8×8 blocking artifacts have been introduced into the sequence. These are illustrated in Figure C.2 which shows the temporal mean and variance images for a 200×200 pel section of the Car Park sequence where there is no motion. The average variance of the pel intensity values in this section is 11 intensity values.

C.2 Car Following Sequence

The Car Following sequence is used in Chapter 4 to test reference block update strategies proposed by the author. This sequence was captured on a digital video camera looking out of the front wind screen of a moving car. The sequence shows a car in front also moving along the road. As a result there are a lot of small, quick movements due to uneven roads which makes the sequence difficult for tracking algorithms. This sequence was captured by Mr Shinichiro Matsunaga, and the author thanks Mr S. Matsunaga for making this sequence available. Example frames from the start, middle and end of the Car Following sequence are illustrated in Figure C.3.

C.3 Moving Photo Sequences

In order to make a quantitative evaluation of the motion estimates, it is necessary to find a test sequence for which the ground truth motion is known. Although this is straight forward

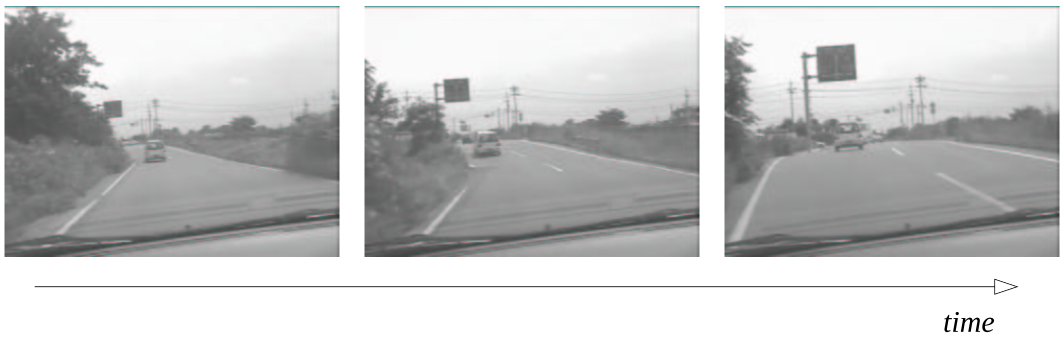


Figure C.3: *Example frames from the start, middle and end of the Car Following sequence.*

for artificial sequences, the ground truth for real sequences is not easily available. The author created the Translating Photo and Advancing Photo sequences for this purpose.

The sequences show a moving photograph, which is an image of a cluttered work bench, captured in the group's laboratory and printed on a colour ink jet printer. The dimensions of the photograph are 14.7cm $\times 10.5$cm. The photograph was fixed to a sheet of card for rigidity. The author constructed a battery powered lego robot which moves in straight lines at approximately 0.02m/s. The photograph was carried by the robot (which is hidden by the photograph) along the desk. In the Translating Photo sequence the motion is from right to left across the desk, in the Advancing Photo sequence the motion is towards the camera. To provide a simple background, a blue dividing screen was placed behind the desk. To minimise shadows and changes in background lighting, lighting was provided by a halogen lamp placed approximately 1.2m above the camera and facing downwards. A plan of the experimental arrangement is shown in Figure C.4.

The sequences were captured using a Sony DCR-TRY890E Digital Video Recorder at 25fps. The automatic focus feature on the camera was used. The images were stored in Sony DV proprietary format, then converted to colour PPM sequences using the commercial image processing packages Adobe Premiere and Adobe PaintShopPro. The Advancing Photo sequence was stored as 360×288 sized images, the Translating Photo sequence was stored as 640×480 sized images which were cropped to 280×200 for processing. Example frames from the two sequences are shown in Figure C.5.

Due to the simple nature of the sequence, the true motion from the scene can easily be calculated directly from the image sequences. This was done by manually tracking the motion of the

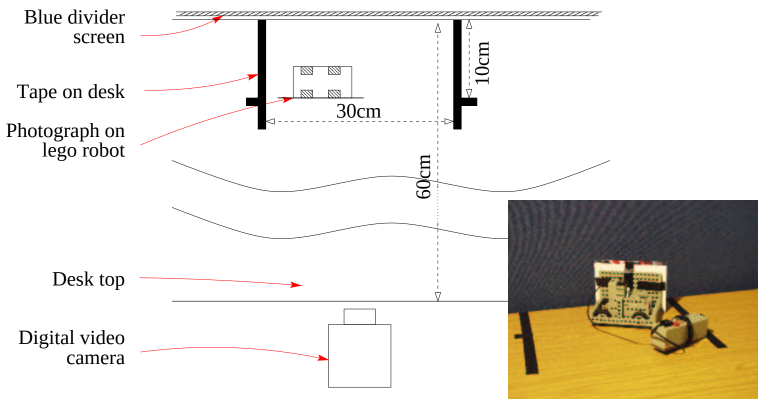


Figure C.4: *Plan showing the experimental set up for capturing the Moving Photograph sequences.*

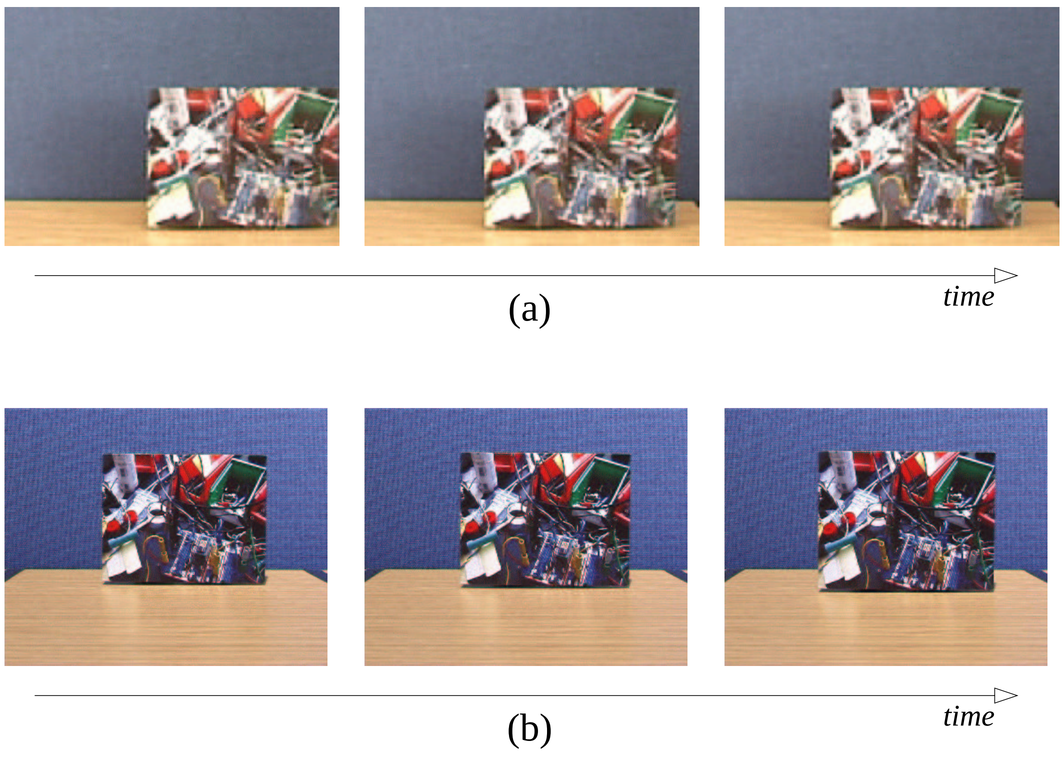


Figure C.5: *Example frames from the start, middle and end of a) the Translating Photo Sequence, b) the Diverging Photo sequence.*

corners of the photograph, then using the resultant motion vectors to calculate the vectors for the remainder of the planar surface. Ground truth was calculated for 12 frames of the Advancing Photo sequence and 41 frames of the Translating Photo sequence. In the Translating Photo sequence, the photo was found to be moving at 0.75 pels per frame. The magnitude of the motion in the Advancing Photo photo sequence was found to be at most 0.78 pels per frame.

The colour image sequences were converted to grayscale (Y) for processing using the following conversion from RGB space [130]:

$$Y = 0.30R + 0.59G + 0.11B \tag{C.1}$$

C.3.1 Moving Checkered Card Sequence

A similar sequence to the Moving Photo sequences is the Moving Checkered Card sequence. This sequence is used in Chapter 3 to illustrate a motion direction estimation technique developed by the author. Instead of a photo, a checkered card is moved horizontally in front of the camera. This sequence was created to more clearly illustrate the performance of the Motion Estimation technique.

Appendix D
Software Implementation

D.1 Introduction

The Motion Estimation and Information Fusion techniques discussed in this thesis were implemented by the author in C++ [163]. C++ was chosen primarily because the language is well known and is easily ported between Unix and Microsoft platforms. Rather than develop code specifically for this thesis, the author has developed a code library that can be easily reused by other researchers working in the field. The encapsulation, inheritance and polymorphism characteristics of object oriented languages make C++ particularly well suited for this purpose.

Sections D.2 to D.5 provides a brief functional description of the key classes underlying the functionality of this library.

In particular, the library was intended for use by the Vision research group in the department of Electronics and Electrical Engineering at the University of Edinburgh. In order to simplify code sharing, the author proposed a set of style guidelines which have been adopted for this library. These guidelines are outlined in Section D.6.

D.2 Key Classes

The software library developed during this thesis was designed to have different classes to represent and implement the different components of a typical image processing system required by the research group. There are three main components to such a system: the images being processed, the algorithm being applied and a mechanism for the input/output of frames. These components are abstracted in the library as the VS_frame, VS_frame_io and VS_image_process classes. The VS_frame, VS_frame_io and VS_image_process classes and the relationships between them are illustrated in Figure D.1.

A fourth class whose functionality was particularly important in this thesis was the VS_frame_vec

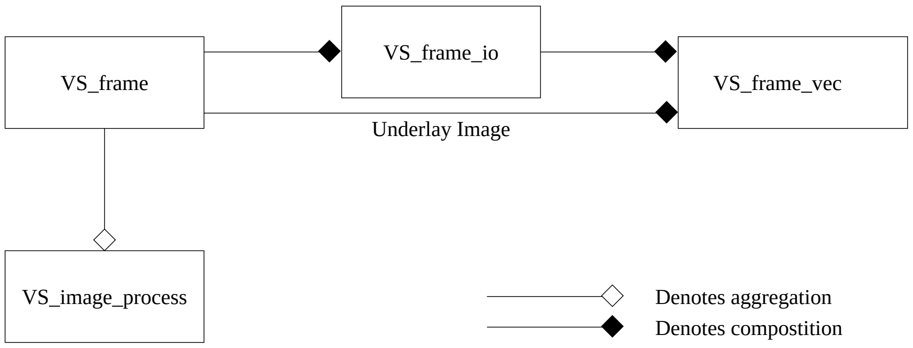


Figure D.1: *The VS_frame, VS_frame_io, VS_image_process and VS_frame_vec classes and the relationships between them.*

class, which provides postscript output of vector flow fields. This class is also illustrated in Figure D.1.

The following sections describe public member functions of the C++ implementations of the VS_frame, VS_frame_io, VS_frame_vec and VS_image_process classes.

D.3 class VS_frame

The VS_frame class is used to store an image and provides functions to access pixels in the image. Images are rectangular arrays of pels. Images can have 1 or more channels, for example an RGB image would have 3 channels and a monochrome image would have 1 channel. Motion vector fields can be stored in 2 channel VS_frame objects.

Pixels are stored as integer (int) values. The decision to avoid real numbers was made because non-integers can not be displayed in PPM format, because floating point arithmetic slows the program speed down, and because where reals are required they can generally be scaled and rounded to integer values with sufficient accuracy.

The copy constructor and assignment operators *are* defined for this class.

VS_frame (int nWidth, int nHeight, int nNumChannels = 1)

Constructor for a frame of width nWidth and height nHeight, with nNumChannels channels. The default number of channels is 1.

VS_frame GetBlock(int nStartX, int nStartY, int nBlckWidth, int nBlckHeight)

Returns the a block of pixels from the image as a new frame. The top left hand corner of the block has image coordinates (nStartX, nStartY) and has dimensions nBlckWidth $\times$ nBlck-Height. A frame with zero pixel values and these dimensions will be returned if an error occurs (eg attempting to access non-existent pixels).

int GetWidth()

Returns the frame width.

int GetHeight()

Returns the frame height.

int GetNumChannels()

Returns the number of channels in the frame.

int GetIntensity(int nX, int nY, int nC = 0)

Returns the intensity value of the nC channel at pixel location (nX, nY). If the pixel location does not exist, returns 0.

int SetIntensity(int nX, int nY, int nVal, int nC = 0)

Sets the intensity value of the nC channel at pixel location (nX, nY) to be nVal. Returns 1 if successful else 0.

D.3.1 Example code

To illustrate the use of some of these member functions, the following example code might be used to implement a threshold operator on a grayscale frame frImage.

```
VS_frame frImage( WIDTH, HEIGHT ); // grayscale frame
int      nX, nY,   // counters
         nThresh;

< code to load in an image, and set the threshold value. >

//
// Look at each pel in turn, set value to 1 if > nThresh, else
// set value to 0.
//
for ( nX = 0; nX < frImage.GetWidth(); nX++ )
    for ( nY = 0; nY < frImage.GetHeight(); nY++ )
        frImage.SetIntensity( nX, nY,
           ( frImage.GetIntensity( nX, nY ) > nThresh )? 0 : 1 );
```

D.4 The VS_image_process class

The VS_image_process class is a base class for all the image processing techniques in the library. The class has just one member function, namely the virtual function Process which takes one VS_frame argument and returns the results of processing as a VS_frame. Image processes derived from this class replace the virtual Process function with one that does a required task.

D.5 class VS_frame_io

The VS_frame_io class was written to enable images to be written to or read from file. The functionality provided by the VS_frame_io class is not provided inside the VS_frame class itself because for majority of frame objects do not require I/O operations in their lifetimes. The class supports both ascii and raw PGM (grayscale) and PPM (colour) file formats.

A VS_frame_io object stores a maximum frame intensity value which is used when displaying frames. If this value is less than the largest pixel intensity in the image, then it will be adjusted when the frame is displayed. This is useful when displaying a frame which has pixels of very low values. For example if all the pixel values in a frame are less than 4, then displaying the frame with max intensity 255 would result in a very dark image with few details. Setting the frame intensity to 0 then allowing the display member function to determine the correct value will give a more meaningful image.

Copy and Assignment *are not* defined for this class.

VS_frame_io(int nMaxIntensity = 255)

Constructor, nMaxIntensity is the maximum intensity of the frame to be output. The default is 255. For example, to display a frame and allow the display function to reset the maximum intensity. the following code might be used:

```
    VS_frame_io( 0 ).DisplayFrame( frImage, ‘‘filename.pgm’’ );
```

VS_frame LoadFrame(char* strFileName)

Loads an image from the file “strFileName” and returns the image as a frame.

int DisplayFrame(frame frImage, char* strFileName)

Displays a frame object frImage in ascii or raw PGM or PPM format to the file “strFileName”. Returns 1 if successful else 0. The chosen output format depends on the number of channels in the frame (1 channel for PGM, 3 for PPM) and on the display mode (see SetDisplayMode). If the number of channels does not correspond to PGM or PPM the display function will fail and return 0.

int MaxIntensity(void)

Returns the current maximum intensity value.

int MaxIntensity(int nIntensity)

Sets the maximum display intensity to nIntensity. If when a frame is displayed any values are greater than nIntensity, then the maximum intensity value will be changed to be the largest value encountered.

int SetDisplayMode(int nMode)

Set the display mode to be ascii or raw PXM. Valid modes are VS_frame_io::APXM and VS_frame_io::RPXM. Returns the last mode if ok else -1.

D.5.1 Example code

A typical use of the VS_frame_io class is to read in a frame for processing by some VS_image_process derived class, then output the results of that processing. This is illustrated by the following example.

```
VS_frame frImage;   // frame to process
VS_image_process impProcess;

//
// Load in the image
//
frImage = VS_frame_io().LoadFrame( ‘‘afile.pgm’’ );

//
// Process the image
//
frImage = impProcess.Process( frImage );

//
// Display the processed image
//
VS_frame_io().DisplayFrame( frImage, ‘‘result.pgm’’ );
```

D.5.2 class VS_frame_vec

When working in the field of Motion Estimation, it is useful to be able to display motion vectors to illustrate motion fields. The VS_frame_vec class provides functionality for displaying vector fields.

int AddVector(int nXPos, int nYPos, int nXVel, int nYVel)

Add the vector (nXVel, nYVel) at pixel position (nXPos, nYPos). Returns 0 if successfull.

int AddVectors(int** pnVectorArray, int nNumVectors)

Add nNumVectors. The vectors are in the nNumVectors by 4 array, where the pixel position for the nth vector is (nVectorArray[n][0], nVectorArray[n][1]) and the horizontal/vertical components of the vector are nVectorArray[n][2]/nVectorArray[n][3].

int ArrowColour(void)

Returns the current arrow colour setting. Currently supported colours are VS_frame_vec::BLACK, VS_frame_vec::WHITE, VS_frame_vec::RED, VS_frame_vec::GREEN and VS_frame_vec::BLUE. The default colour is VS_frame_vec::BLACK.

int ArrowColour(int nColour)

Set the a new arrow colour. Returns the old arrow colour if successful, else returns -1. See ArrowColour(void) for valid arrow colours. Only supported for EPS output.

int ArrowHeadLn(void)

Returns the current arrow head length. See ArrowHeadLn(int nLength) for details.

int ArrowHeadLn(int nLength)

Set the a new arrow head length. Returns the old length if successful, else returns -1. The Arrow head is an isosceles triangle whose base width is 3/4 of the height. Only supported for EPS output.

int ArrowWidth(void)

Returns the current arrow width. See ArrowWidth(int nWidth) for details.

int ArrowWidth(int nWidth)

Set the a new arrow width. Returns the old width if successful, else returns -1. The width is used for the arrow line. Only supported for EPS output.

int ClearVectors()

Reset the vectors all to 0.

void Display(char* filename)

Display the vectors to the file “filename”.

int DisplaySmallVectors(int nVal)

Call with nVal non-zero to allow vectors which are smaller than the arrow to be drawn (in which case only the arrow head will be seen). Returns the last value of nDisplaySmallVectors.

int SetFrameDims(int nXVal, int nYVal)

Set the frame dimensions to height nXVal and width nYVal. Returns 0 if successfull. The default frame size is CIF (312 by 288). If an underlay image is used, then the underlay and display dimensions must be the same.

int FrameHeight(void)

Returns the frame height in pixels.

int FrameWidth(void)

Returns the frame width in pixels.

int MaxDisplayLength(void)

Returns the maximum length of displayed vectors.

int MaxDisplayLength(int nVal)

Set the maximum length of displayed vectors, must be at least 1. Returns old display length if successfull else returns -1.

int MaxIntensity(void)

Return the maximum intensity of pixels in an underlay image.

int MaxIntensity(int nVal)

Set the maximum length of pixels in an underlay image, must be at least 1. Returns old maximum intensity if successfull else returns -1.

int MaxVectorLength(void)

Return the longest vector length.

int MaxVectorLength(int nVal)

The longest vector length will usually be the longest vector encountered so far. This function sets the longest vector length to nVal (must be at least 1), which can be helpful for looking at very small vectors in a field containing large vectors. Returns the old maximum vector length if successful else returns -1.

int SetDisplayType(int nType)

Set the output format to pgm (nType = frame_vec::PGM) else to encapsulated postscript (nType = frame_vec::PS)

int SpacingFactor(void)

Returns the spacing factor.

int SpacingFactor(int nVal)

For pgm output, each pixel can be scaled up nVal (¿0) times to make it easier to see dense vector maps. This is similar to zooming in on the image. Returns the old spacing factor if successfull else 0.

int Underlay(frame frUnderlay)

Sets the frame frUnderlay to appear under the displayed vectors. Must have same width/height as the frame_vec object. Single channel frames only supported. Returns 1 if successfull else 0.

D.6 Coding Style

ISG Vision is a research activity at the Department of Electrical Engineering at the University of Edinburgh. To improve group efficiency and encourage discussion, code and utilities are shared between members. To enable code sharing to be effective, common practices have been adopted. One aspect of this stringly supported by the author has been the adoption of a uniform coding style, which makes it easier for one member to understand the code written by another. The purpose of this appendix is to document this style.

The purpose of this document is not to discuss good programming practices, and it's scope is limited mainly to considerations of readability. Nonetheless, adhering to a single style and giving clear comments is a good first step towards good programming.

D.6.1 Comment style

Comments are mainly used to explain code, but can also be used to improve clarity of the code layout. Templates showing comment the layout for source files have been written, these are included at the end of this appendix.

The double-slash (//) comment is preferred over the slash-asterix (/* */) comment style.

Comments on Functions

A comment at the start of a function describes the purpose of the function and clarifies what the function arguments should be (if this is not made clear by the argument names and types). The style for function comments is given later, with an example.

Comments on Blocks of Code

Comments explaining blocks of code should start and end with a comment line consisting only of the initial double slash. This makes comments clearer and helps distinguish blocks of code.

Small Comments

Where a comment is used just to explain a single line of code, for example to clarify the purpose of a variable, the comment can appear at the end of the line.

Example

The following example illustrates these points:

```
//
// Comment explaining the following block of code
//
for ( nX = 0; nX < 100; nX ++ )
{
    < code >

    nValue *= 2; // Comment explaining this line

    < code >
}
```

A single line of code can often have sufficient importance to justify the longer comment form.

'Commenting out' code should not be done using comments(!), instead pre-processor operatives should be used, for example:

```
< code >

#if 0
< commented out code >
#endif
```

D.6.2 White Space

White space generally makes code easier to read, and so should be used generously. Space can be included horizontally and vertically.

Horizontal spacing

Examples of where spaces should appear include before and after operators, after commas in argument lists and before and after parentheses. One important exception is in function calls, where syntax requires the opening parenthesis to directly follow the function name. The following line of code illustrates these points:

```
int nValue += (int) rint( ( flValue + 10 ) / 3.142 );
```

Without the spacing, this line would look like this:

```
int nValue+=(int)rint((flValue+10)/3.142);
```

Vertical spacing

Vertical spacing can be helpful in distinguishing one block of code from another, and so the use of empty lines to separate blocks of code is encouraged. The comment and bracing styles discussed in this document are preferred because they encourage the use of a lot of vertical space.

Long lines of code should be split over a number of lines. Where convenient points for indenting successive lines exist, such as an '=' or a parenthesis, they should be used. Otherwise, the extra lines should be indented at least two spaces further than the first line. For example, the addition in the following code has been split over two lines:

```
for ( nX = 0; nX < QuadEl.nWidth; nX++ )
     for ( nY = 0; nY < QuadEl.nHeight; nY++ )
         flMean += frImage.GetIntensity( nX + QuadEl.Pos.nXPos,
                                         nY + QuadEl.Pos.nYPos );
```

This is far clearer than the following, which should be avoided:

```
for ( nX = 0; nX < QuadEl.nWidth; nX++ )
    for ( nY = 0; nY < QuadEl.nHeight; nY++ )
        flMean += frImage.GetIntensity( nX + QuadEl.Pos.nXPos, nY
+ QuadEl.Pos.nYPos );
```

No line should have a length greater than 80 characters, since this causes problems for printers and code should be as readable on paper as on screen.

D.6.3 Indents and Braces

The BSD style for positioning braces, where braces are placed on separate lines, is preferred. This style is illustrated here:

```
if ( condition )
{
    <body>
}
```

Code should be indented from the left margin to indicate nested block depth. The indents are 4 spaces long.

D.6.4 Naming Conventions

Choosing appropriate names for classes, variables, constants, functions and even filenames helps other programmers to understand the code. This section discusses the naming conventions adopted by the Vision Systems group.

Variables

Variable names should describe both the type and the purpose of the variable. Variable names comprising a number of words should be made more readable either by using capital letters at the start of each word or by using underscores between words. For example:

```
int nVeryLongVariableName = 0;
int nvery_long_variable_name = 0;
```

The choice of style is not prescribed by this document, but as always, programmers should ensure that the chosen style is consistent inside files.

The first few characters of variable names should be used to indicate the type of the variable. The following prefixes are used by the group:

c	char
fl	float
db	double
n	int
p	pointer
pc	char *
pn	int *
str	char *

For example, a suitable name for an array of integers representing the sum of some vector list might be "pnSum". str- is used rather than pc- to suggest a text string.

Some style guides suggest that objects should have prefix 'o', this is not advocated by this document, since pretty much any variable can be interpreted as an object.

Constants

Hard-wired constants, such as a value for π, should should be in capitals. Immutable class attributes which are initialised by the class constructor have the same name conventions as mutable variables.

Constants should be declared of type 'const' rather than #define'd, since this enable the compiler to perform better type checking. For example:

```
const double PI = 3.142;
```

Functions / Classes

The choice of naming style for private functions and classes is not prescribed by this document. It is very important, however, to maintain a uniform feel to the API, so the style of classes and functions forming part of the API is prescribed.

Classes intended to form part of the Vision Systems group library should start with "VS_", the remaining characters should be in lower case with long names split up by underscores.

Public member functions of these classes should use capital letters to split up long names. For example, the following member functions come from the VS_frame and VS_frame_vec classes:

```
      void VS_frame::SetIntensity( ... );
      int  VS_frame::GetIntensity( ... );

      int  VS_frame_io::MaxDisplayLength( ... );
```

Accessing Attributes in the API

Where possible, functions should be provided to allow access to object attributes, rather than allowing the attributes to be changed directly. This allows greater scope for validation.

D.6.5 Code Templates

Figures D.2 and D.3 show templates for C++ source and header files respectively.

An important role of the comments at the top of the file is to indicate the authors of the code and any modifications. This helps other group members who have questions about the code. Modifications to a file after creation should be tagged by the programmers initials, for example as follows:

```
//
// file    : VS_program.cc
// author  : Andrew Mark Peacock
// created : 26 MAY 2001
//
// Code implementing the class VS_program.
//
// Modifications :
//
// MR   Mike Robinson
//
// 10 JUN 2001   MR   Added SetVariable member function
//

//
// (c) 2001 Vision Activity (ISG), the University of Edinburgh
//

< code >

///////////////////////////////
```

```
// member function: VS_program::SetVariable
//
// MR added this function to set the variable.
//
///////////////////////////////
```

```
//
// file    :
// author  :
// created : DD MMM YYYY
//
// Description
//
// Modifications :
//

//
// (c) 2001 Vision Activity (ISG), the University of Edinburgh
//

////////////////////////////////////////////////////////////////////////////////
// Include files

//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
// Local Defines

//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
// Local Function Declarations

//
////////////////////////////////////////////////////////////////////////////////

///////////////////////////////
// constructor function:
//
///////////////////////////////

///////////////////////////////
// destructor function:
//
///////////////////////////////

///////////////////////////////
// assignment function:
//
///////////////////////////////

///////////////////////////////
// copy function:
//
///////////////////////////////

///////////////////////////////
// member function:
//
///////////////////////////////

///////////////////////////////
// non-member function:
//
///////////////////////////////

// End of File
////////////////////////////////////////////////////////////////////////////////
```

Figure D.2: *Template for a C++ source (.cc) file.*

```
//
// file    :
// author  :
// created : DD MMM YYYY
//
// Description
//
// Modifications :
//

//
// (c) 1999 Vision Activity (ISG), the University of Edinburgh
//

#ifndef __TEMPLATE_H__
#define __TEMPLATE_H__

////////////////////////////////////////////////////////////////////////////////
// Include files

//
////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////
// Local Defines

//
////////////////////////////////////////////////////////////////////////////////

///////////////////////////////
// class
//
//////////////////////////////
class  :
{

private:

protected :

public :

    //
    // Constructor
    //

    //
    // Destructor
    //

    //
    // Copy
    //

    //
    // Assignment
    //

};

////////////////////////////////////////////////////////////////////////////////
// Inline functions

//
////////////////////////////////////////////////////////////////////////////////


#endif //__TEMPLATE_H__

// End of File
////////////////////////////////////////////////////////////////////////////////
```

Figure D.3: *Template for a C++ header (.h) file.*