Optical Flow Estimation Using Steered-L^1 Norm

Ammar Zayouna

School of Science and Technology

Middlesex University

This thesis is submitted to Middlesex University in partial fulfilment of the requirement for the degree of

Doctor of Philosophy

October 2016

I would like to dedicate this thesis to ‘Ezzat Zayouna’ and ‘Nadia Elia’ my loving parents.

Thank you both.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Ammar Zayouna
October 2016

Acknowledgements

I would like to thank my director of study ‘Prof. Richard Comley’ for his invaluable guidance, continuous support and encouragement throughout the duration of this research. This work was not possible without him. I would also like to thank my second supervisor ‘Dr. Daming shi’ for his help in this thesis.

Abstract

Motion is a very important part of understanding the visual picture of the surrounding environment. In image processing it involves the estimation of displacements for image points in an image sequence. In this context dense optical flow estimation is concerned with the computation of pixel displacements in a sequence of images, therefore it has been used widely in the field of image processing and computer vision. A lot of research was dedicated to enable an accurate and fast motion computation in image sequences. Despite the recent advances in the computation of optical flow, there is still room for improvements and optical flow algorithms still suffer from several issues, such as motion discontinuities, occlusion handling, and robustness to illumination changes. This thesis includes an investigation for the topic of optical flow and its applications. It addresses several issues in the computation of dense optical flow and proposes solutions. Specifically, this thesis is divided into two main parts dedicated to address two main areas of interest in optical flow.

In the first part, image registration using optical flow is investigated. Both local and global image registration has been used for image registration. An image registration based on an improved version of the combined Local-global method of optical flow computation is proposed. A bi-lateral filter was used in this optical flow method to improve the edge preserving performance. It is shown that image registration via this method gives more robust results compared to the local and the global optical flow methods previously investigated.

The second part of this thesis encompasses the main contribution of this research which is an improved total variation L^1 norm. A smoothness term is used in the optical flow energy function to regularise this function. The L^1 is a plausible choice for such a term because of its performance in preserving edges, however this term is known to be isotropic and hence decreases the penalisation near motion boundaries in all directions. The proposed improved L^1 (termed here as the steered-L^1 norm) smoothness term demonstrates similar performance across motion boundaries but improves the penalisation performance along such boundaries.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ofc Optical Flow Constraint

AAE Average Angular Error

AEPE Average End-Point Error

C2F Coarse-to-Fine framework

CT Computed Tomography

CUDA Compute Unified Device Architecture

DIS Digital Stabilisation Systems

EC2F Extended Coarse-to-Fine framework

EIF Extended Intermediate Filtering

FED Fast Explicit Diffusion

FOE Focus-of-Expansion

FPGA Field Programmable Gate Array

GPU Graphic Processing Unit

HOG Histogram of Oriented Gradients

KITTI Karlsruha Institute of Technology vision benchmark suite.

MATLAB Matrix Laboratory

MRF Markov Random Field

NNF Nearest Neighbour Field

OpenCV Open Source Computer Vision

QPBO Quadratic Pseudo Boolean Optimisation

RANSAC Random Sample Consensus

SfM Structure-from-Motion

SIFT Scale-Invariant Feature Transform

TTC Time-to-Contact

TV Total Variation

UAV Unmanned Aerial Vehicle

VLSI Very Large Scale Integration

Chapter 1

Introduction

Optical Flow is a low level image processing task that has a wide range of applications in many areas across computer vision. In its basic notion, optical flow can be defined as the computation of image points displacements over time [1], [2]. The displacements of image points that are computed in the 2D image domain correspond to the relative motion between the camera and the objects points in the real world, as well as among the objects themselves. This gives an important cue in computer vision and can be used in many tasks such as Structure-from-Motion, image registration, object tracking, and robot navigation (see Section-2.3). In the literature there have been many algorithms developed for the purpose of computing optical flow fields. In this thesis the focus is on what is known as the variational methods, which use variational calculus to find optical flow displacements fields. Variational methods are perhaps the most popular methods to calculate optical flow due to its sub-pixel accuracy and the fill-in effect that permits the ability to find optical flow at image locations even if no or limited image information is available [3], [4].

Since the marquee paper of Horn-Schunck [1], which used variational methods to find optical flow, there has been much research dedicated to improve the performance of optical flow computation, yet the main issues facing computing the optimum optical flow field still remain. Examples of such issues are robustness to illumination changes, motion discontinuousness and occlusion, real-time computation, small structures and large displacements and texture-less areas. Among those problems preserving motion boundaries while estimating optical flow field has also attracted a lot of interest since the early days of optical flow consideration. In addition to advances in optical flow algorithms and the way they are understood, another important aspect of optical flow has attracted attention, that is the means of measuring the accuracy of the computed displacement field and datasets to test these al-

gorithms. Some examples of error measures are Average Angular Error (AAE), Average End-point Error (AEPE) and Interpolation Error (IE), in addition to flow colour-coding to qualitatively analyse the displacement fields [5], [6].

Figure-1.1 illustrates the concept of optical flow, an object at position A moves towards position $A^{'}$ in time dt. Optical flow algorithms aim at computing the velocity at which each point in the image domain moves, where the velocity is decomposed into two components in dx/dt, dy/dt in both the x and y directions. This information can be extracted from an image of the object at position A and time t, and an image of the object at position $A^{'}$ and time $t+1$. It follows that $dt = (t+1) - t$.

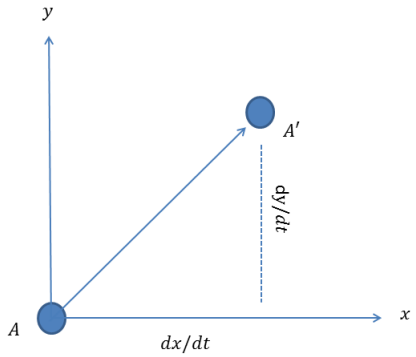


Fig. 1.1 Point velocities.
The aim of optical flow is to find the velocity of the point A which moves to location $A^{'}$ in the second image.
If dt is considered to be equal to unity then the problem comes to down to finding the displacement from A to $A^{'}$.

These images can also be taken from a sequence of video frames, if dt is considered to be equal to unity, then the problem comes down to finding the displacement of image points.

The displacement field can be represented as a vector originating from points in the image heading towards the destination points in the second image. In this context it is useful to distinguish between two concepts, these are non-dense and dense optical flow fields. The non-dense flow field depicts displacements at discontinuous points of the image sequence. The dense flow field on the other hand depicts the displacements for most points of the image. Hence image points are actually the pixels of these images. The density ratio of the calculated flow field differs for various methods.

Consider the following two images taken from the Middlebury dataset [6] in Figure-1.2. The images exhibit relative movement between the camera and the objects, from the first image taken at time t to the second image taken at time t+1.

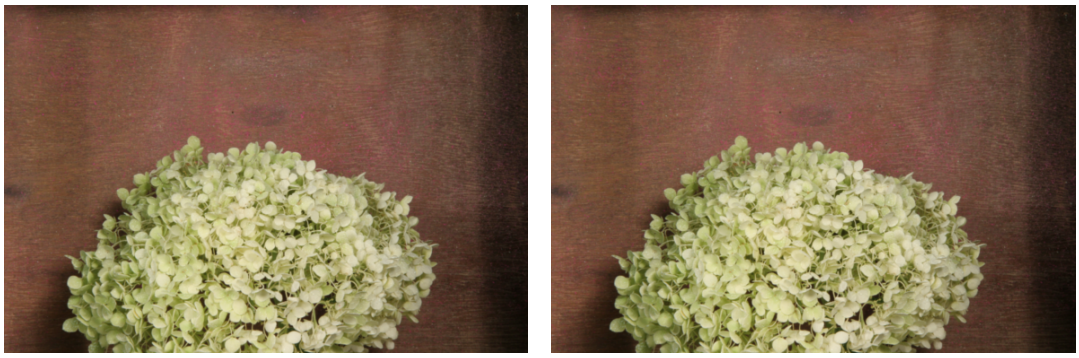


Fig. 1.2 Image sequence 'Hydrangea' [6].
Left: Image taken at time t. Right: Image taken at time t+1.

Figure-1.3 depicts the computed vectorised displacement field.

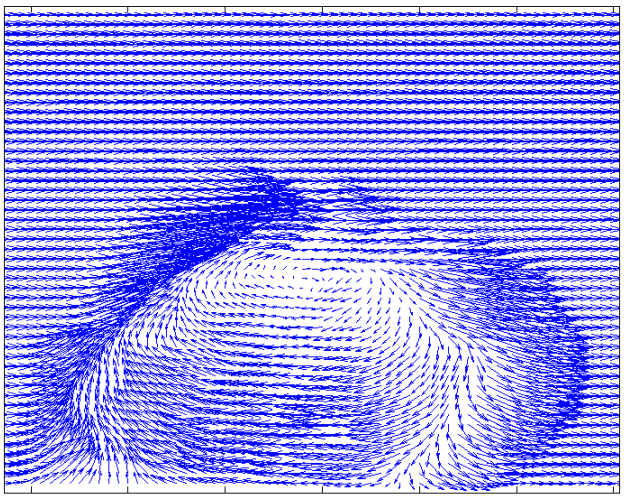


Fig. 1.3 Vectorised visualisation of optical flow field.

In Figure-1.3 the displacement fields are depicted as arrows originating from various points of the first image of the sequence. The heading direction of each arrow depicts the direction of the point motion, while the length of these arrows are proportional to the magnitude of the movement.

The aim of optical flow algorithms is to calculate two components for the displacement field in the x and y directions. A colour-coded flow field can be obtained which helps to visually

inspect the calculated displacement field. This way of visualising the flow field is more suitable for dense displacement fields. Throughout this thesis the colour coding shown in Figure-1.4 is used to visualise the displacement fields [6]. Although colour-coded optical flow field is not a precise way to judge the efficiency of the optical flow algorithm, it is useful to give a general impression on the results, especially when no quantitative measures and ground truth data are available. In this colour-code the direction of displacements is coded by the hue, and the magnitude of the displacements is coded by the saturation. For example if a pixels moves in the upper left direction of the image, the pixel appears in blue. The more saturated colour indicates bigger displacement. In other words the colours correspond to the angle and the magnitude of displacements. The colour code in this thesis follows the colour code of the Middlebury dataset[1]. The visualised flow field results are computed by applying the colour and saturation corresponding to the magnitude and orientation of the displacements.

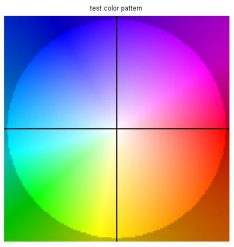


Fig. 1.4 Colour code visualisation for optical flow displacement fields.
The direction of displacements is coded by the hue, and the magnitude of the displacements is coded by the saturation.

Based on the ground truth optical flow field of the image sequence set, the displacement field of Figure-1.2 can be visualised as follows:

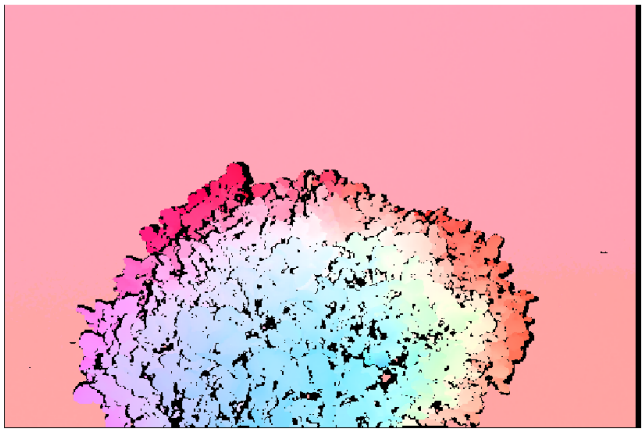


Fig. 1.5 Ground truth for ‘Hydrangea’.
This visualisation follows the colour code in Figure-1.4.

[1] http://vision.middlebury.edu/flow/data/

This thesis studies several aspects and applications of optical flow. In the next section, the motivation of the work in this thesis is discussed. The is followed by sections introducing several concepts and applications related to optical flow, and demonstrate its importance in the field of computer vision. At the same time, several limitations are discussed, and possible solutions are proposed which will be introduced later in this thesis.

1.1 Motivation

Motion is an important cue in the world around us. As part of the endeavour to perceive the surrounding environment, living creatures need to observe the relative motion between objects in the environment and the observer. In static environments it is enough to only identify the various objects surrounding the perceiver in order to interpret the scene. This is done by observing the different colours, textures and locations of these objects. As soon as relative motion between the objects and the perceiver takes place additional information is available, and the objects' features are not enough to form a complete idea about the environment. The relative motion between the observer and the surrounding environment can help to carry out several tasks, for example avoiding collision and tracking and estimating objects' distances. Analogues to this is machine and computer vision. Motion is also an important cue which enables these artificial systems to interact with their surroundings. Using the estimated motion several useful systems can be designed such as obstacle avoidance, objects tracking, path planning and steering.

In the heart of motion estimation systems is the optical flow computation, where motion can be estimated from scenes in an image sequence. Estimating motion via sparse descriptors has been successful for some applications. On the other hand dense optical flow that attempts to find the displacement for each point (pixel) in the image sequence offers richer information. Optical flow has been utilised in many computer vision applications, some examples are autonomous navigation and object tracking. Estimating dense optical flow is non-trivial, and research in this area is still attracting a lot of attention to enhance existing algorithms.

This research investigates the topic of optical flow, with an emphasis on the dense optical flow using global methods which were shown in previous research to give good results. During the course of this research several shortcomings and issues of optical flow computation are addressed and solutions are proposed. Although this research does not propose an ultimate solution for the problem of optical flow computation, it certainly offers an improvement in several aspects. The algorithm presented in this thesis is a developmental method,

and the improvement in estimating of optical flow is demonstrated using quantitative and qualitative results. The results show an improvement when compared with other methods sharing similar principals. This research concludes with recommendations for further research and suggestions to be investigated in the future.

1.2 Image Registration

Image registration has been widely used in many applications such as change detection and monitoring, image mosaicing, remote sencing, and medical imaging [7], [8]. Image registration is the process of matching (aligning) two or more images, these images are either from different viewpoints or using different sensors. In this case the image registration is called multi-modal. Alternatively, images can be taken for the same scene but at different points in time, in this case image registration is referred to as mono-modal [7]. Usually one of the images is considered as a target image, while the other is a source image which is required to be aligned. Based on the previous definition, the relation of image registration and optical flow can be easily established in the mono-modal case. Optical flow can be considered as a special case of mono-modal image registration, where the deformations or the changes between the two images are relatively small [7]. In the case of images with rigid contents, image registration based on feature detection and matching is widely used. This type of registration is also referred to as parametric image registration, where the parameters of the mapping or transformation function are estimated from these features. On the contrary, if image contents undergo deformations and changes, deformable image registration methods such as the methods based on optical flow becomes more suitable. This is due to the dense displacements field which can describe the transformation in a detailed way.

To ensure an accurate registration, accurate optical flow displacements should be found. Usually this is not an easy process, especially when the image sequece encompass noise which renders the optical flow estimation inaccurate. Local and global optical flow methods were used in optical flow based image registration [9], [10]. As it will be shown later, the local method assumes that correspondence can be established for small neighbourhoods between images, thus these methods are relatively robust. Global methods on the other hand try to establish these correspondences between single pixels rather than neighbourhoods, thus dense results can be obtained. This leads to better image registration results since more detailed information is included. However the global methods are known to be prone to noise.

To demonstrate the affect of noise on the registration results, consider the following sequence of images in Figure-1.6. In this image a zero mean Gaussian noise with variance of (0.005) was added, the optical flow displacement field is estimated via the Horn-Schunck method [1] with implementation described in [11]. The flow field is then used to register the original second image (i.e. without noise contamination).

Fig. 1.6 The first frame (frame-10) of Walking sequence [6].
Left: Original frame. Right: Contaminated with noise.

The registered image is shown in Figure-1.7 next, where it can be seen that the registered image has lost some of its sharpness and became blurry. Any further smoothing would introduce more blurriness. The blurriness can be noticed in areas with small details and also straight lines.

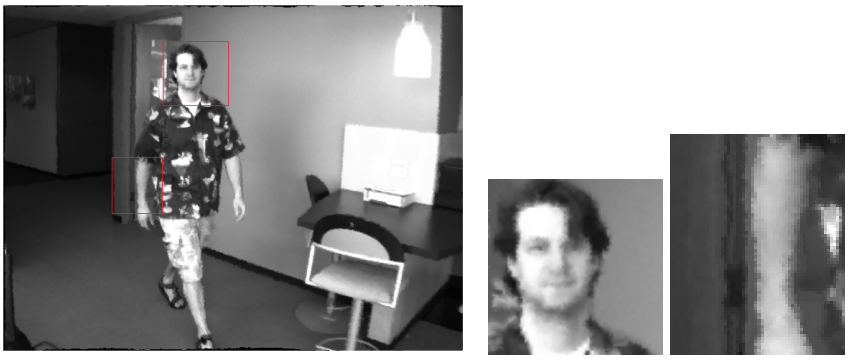


Fig. 1.7 Registered image.
Left: Registered image. Middle and Right: Parts of the image to illustrate the blurry results of image registration.

A combined local-global (CLG) method was proposed by Bruhn et al. [12], combining

benefits of both local and global methods, thus it performs better in terms of accuracy and robustness. Both the local and global methods were used for image registration for different purposes [9], [10]. However, the CLG version of optical flow has not been considered for image registration problems. In Chapter-3 a method for image registration based on CLG optical flow is presented, with the aim of improving the performance of image registration. The CLG algorithm proposed in [12] can be expressed in terms of a linear structure tensor, this renders the calculated optical flow blurry across motion boundaries. To address this problem a CLG optical flow based on a bilateral structure tensor is presented and used here. It will be shown that the proposed modification improves optical flow quality across such boundaries. Although the main field of application for non-parametric image registration is medical imaging, the focus of this thesis is the general concept of this type of image registration. Therefore the results of the algorithm are displayed using general purpose test images that are not necessarily medical. However further investigation on the use of CLG optical flow in medical imaging can be considered for future work. Chapter-3 introduces the theoretical background for optical flow, and lays the foundation of common techniques that are used in algorithms in the rest of this thesis.

1.3 Motion Boundaries Conservation and the Filling-in Effect

Despite the recent advances in algorithms estimating optical flow fields, they still suffer from several issues. The techniques and algorithm are usually tailored to address a certain problem or problems, at the same time it may affect other aspects of the calculations. For example the algorithm proposed in [13] aimed at detecting large displacements of small objects in the image by extending the calculation to include sparse descriptors. Although the algorithm succeeded in this to an extent, the overall accuracy of the calculation showed a small decrement. Edge conservation is among those issues and was paid a lot of attention to in the research of optical flow algorithms. One can imagine that knowing the exact motion boundaries are crucial in many applications, additionally the exact localisation of the motion boundaries increases the accuracy of the calculated optical flow field. If the motion boundaries are not well preserved in the calculation, it appears as a blurry edge between two smooth optical flow regions.

The blurred edges and boundaries in the global optical flow algorithms can be attributed to many reasons, one of the main reasons is the smoothness term. In the calculation of optical

flow the displacement field is assumed to be smooth across the whole image, in other words the displacement of neighbouring pixels have very close values. In reality neighbourhood pixels may belong to different objects, each of which exhibit dissimilar motion, so the pixels will no longer be very close to each other in the next image. If the smoothness term strongly penalises this diverse movement, the motion boundaries between those pixels becomes blurry. For example in the marquee work of Horn-Schunck [1] a quadratic penaliser was used. This penaliser is known to penalise the displacement field severely. Consider the 'Urban2' image sequence taken from the Middlebury dataset, Figure-1.8 next demonstrates the visualisation of the displacement field for this sequence. A modern implementation of this algorithm was presented and described in detail by Sun et al. [11]. The code was downloaded from their website[2], the original parameters of that code is kept unchanged. This code is used to generate the following optical flow field.

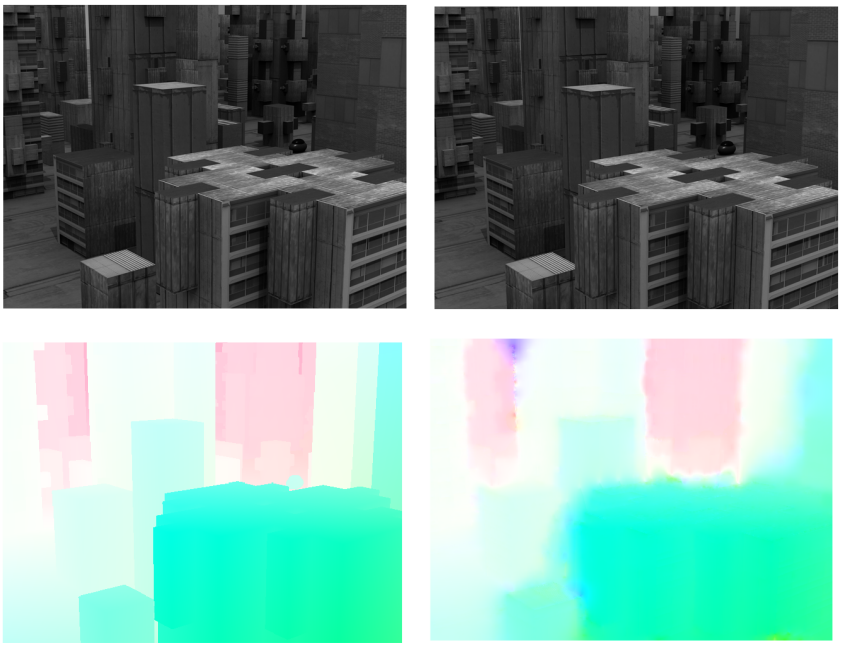


Fig. 1.8 Image sequence and optical flow for 'Urban2'.
Top left: First image. Top right: Second image. Down left: Ground Truth. Down right: Estimated optical flow using Horn-Schunck algorithm.

It can be seen by examining the previous figure the blurriness which makes it difficult to identify and distinguish the different objects' motion in that image sequence. The problem of edge blurriness may be found more in what is known as *'Global methods'* which requires

[2] http://cs.brown.edu/~dqsun/research/software.html

the use of a smoothness term. Further discussion on global methods and its counterpart *'Local methods'* is found in Section-3.5.

One way to improve the edge conservation performance is to retract from the assumption of smoothness across an image, and assume a *'Piecewise Smooth'* flow field instead. Indeed the piecewise smoothness can better characterise the displacement field, and hence it can improve the estimation of the flow field. An example for a piecewise smoothness function is the total variation L^1 norm, which was used in several algorithms as the smoothness (Regularity) term [14], [9], [4].

The L^1 norm provides an isotropic penalisation for the displacement field, i.e. a penalisation regardless of the direction of the flow. The regularisation term in the global optical flow methods is responsible for what is known as the *'Filling-in'* effect, where displacements of pixels with no information can be estimated via the information contained in the neighbouring pixels. This is a very important property of the global optical flow methods. The filling-in effect is induced by the penalisation of the regularity term. A piecewise smooth regularity term reduces penalisation at locations near motion boundaries to permit discontinuities. Since the L^1 norm is an isotropic function as pointed out earlier, the filling-in effect reduces in all directions near motion boundaries, reducing the accuracy of the optical flow algorithm.

Some researchers proposed to adapt penalisation with the direction of image structures, however this may produce over segmentation in the estimated flow field [4], [15]. In this thesis an improved L^1 norm penalisation is introduced, which is called '*steered-*L^1' smoothness because the penalisation direction is adapted to image structures, while penalisation magnitude rely on the flow field value. The detailed discussion on the isotropic and the new anisotropic L^1 smoothness term is introduced in Chapter-4.

1.4 Thesis Outline and Research Questions

This thesis investigates optical flow computation in several aspects, with a focus on motion boundaries preservation. The structure of this thesis is as follows: Chapter-2 provides a literature review for optical flow and related areas, in particular a discussion of the main elements and advances of the variational optical flow field as well as the main problems that have been addressed in research over the previous years. Additionally this chapter touches on the applications of optical flow. The focus of this chapter is on the variational algorithms, and starts with a brief introduction to the general notion of optical flow, and the

general energy function used to compute displacements. Various subsections are dedicated to discuss the advances of the main terms of the variational formulation of optical flow, namely the data and smoothness terms.

Chapter-3 investigates a particular algorithm for computing optical flow, namely the Combined Local-Global (CLG) optical flow [12]. This chapter also discusses the topic of image registration, and examines the relation between image registration and optical flow estimation. It will be shown in this chapter that the use of CLG optical flow for image registration yields dense and more robust results. This is in comparison to other image registration techniques which are based on either local or global methods. This chapter starts by introducing the theoretical background and concepts used in calculating optical flow, which are generally used in most of variational optical flow algorithms. After that the CLG optical flow for image registration with some initial results obtained at early research stages is demonstrated. This chapter also introduces an improved version of the CLG method using bi-lateral filters, this improves edge-preserving and accuracy of displacement field estimation. The main research questions that this chapter addresses are:

- How would the use of CLG optical flow methods improve image registration and motion estimation?
- How is it possible to improve the edge preserving performance of CLG optical flow?

In Chapter-4 a steered-L^1 norm smoothness term is proposed, this smoothness term renders this function anisotropic with respect to the direction of image structures. However the magnitude of the penalisation is adapted according to the flow field which circumvents over segmentation of the optical flow field. Additionally, this chapter includes a discussion on different smoothness terms and their influence on the edge preserving performance of optical flow algorithms. This chapter starts by introducing the total variation $TV-L^1$ algorithm used to calculate optical flow, then uses these concepts to propose the algorithm. In addition to that, a formulation for the use of a delayed linearisation data term is introduced with the use of image gradient to improve robustness against illumination changes. This chapter addresses the following questions:

- How can the L^1 smoothness be modified to become anisotropic and does not impede the filling-in effect near edges? and;
- How to do this modification despite the fact that this function is not continuously differentiable?
- How can a non-linearised (delayed linearisation) data term be used in a primal-dual

optical flow algorithm? and;

- Does using such a robust non-linearised data term with image gradient render the performance better compared to other methods?

Chapter-5 is dedicated to the experiments conducted for the proposed algorithms. It is divided into two parts, the first part is concerned with experiments on image registration, with a focus on showing the difference in registering quality between the global optical flow methods [1] and the proposed modified CLG optical flow. Results in this section extends the preliminary results demonstrated in Chapter-2, and uses the Middlebury dataset [6]. The second part is concerned with experiments on the proposed optical flow algorithm with the steered-$L1$ norm, and demonstrates the improvements and how it affects the results. Experiments are conducted to show the different effect of main constituent of the algorithm. The Middlebury dataset also used for this purpose.

Chapter-6 concludes this thesis. In this chapter the work presented is assessed. Improvements are highlighted and shortcomings are pointed out. A detailed discussion on future improvements is included in this chapter, in addition to possible future research directions.

This thesis also includes two appendices. The first appendix (Appendix-A) includes some preliminary work. The proposed algorithm in Chapter-4 is supplemented with a descriptors' matching term to enable the detection of small objects in the image sequence, this work is part of a future research direction that will be investigated and has application in the field of robot navigation. Appendix-B includes a paper that was published containing the work presented in Chapter-3.

Published results

The results obtained in this thesis have been published in two papers to date. The work presented in Chapter-3 is published in a paper which can be found in Appendix-B. The work in Chapter-4 is published in a paper accepted for presentation at the '24th. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2016 (WSCG 2016)'.

Chapter 2

Literature Review

This chapter introduces a literature review for the development of optical flow computation. This review focuses on variational algorithms because they are the main topic of the thesis. The introduction of this chapter includes a general discussion and explanation for the topic of optical flow. It also introduces the general formula for optical flow computation. The rest of this chapter is divided as follows: in the next section a discussion on the data term of the general optical flow energy function is presented. In the following section the smoothness (regularity) term of that equation is discussed. In the final section a review of the main applications of optical flow, and a demonstration on how this technique is used in image processing and computer vision fields is presented. Formulas and equations will also be included in this chapter for the purpose of further illustrations of ideas.

Optical flow aims at finding displacement fields for pixels between two images taken for the same scene at time t and $t+1$. It has been used widely in many computer vision applications, such as mobile robotics, structure-from-motion, object tracking, etc. Although the computation of optical flow has been widely researched, and many problems have been addressed, solutions are far from being perfect, and algorithms still need further investigations and improvements. Some examples of common problems in optical flow calculations are small scale with large displacements, discontinuity (edge) preserving, texture-less areas, etc.

Algorithms for finding optical flow fields can be classified in many ways. An early taxonomy for optical flow algorithms was reported by Barron et al. [5]. They classified these methods into 4 main groups, namely: region-based matching, energy-based, phase-based, and differential. In addition to that they also compared the performance of these methods. In region-based methods, algorithms try to find block matches between search areas in the

two images. This is done by maximising (minimising) a similarity (dissimilarity) measure. The displacements found usually lack sub-pixel accuracy, therefore an additional refinement step is needed. Examples for such algorithms are [16], [17], [18]. Energy-based methods work in the frequency domain. An example for such algorithms is the work presented by Heeger in [19]. He used Gabor filters, and used these spatio-temporal filters to extract optical flow. The phase-based algorithms were denoted by that name in the taxonomy because they rely on the phase behaviour of band-pass filters. An early attempt in this field was the work of Fleet-Jepson [20]. The authors used *'local phase gradient'* constancy assumption to estimate component velocity. The phase-based methods are computationally demanding. A relatively recent work was presented by Pauwels et al. [21]. The authors in the latter paper attempted to parallelise the algorithm of Fleet-Jepson ona CUDA platform, additionally they used Coarse-to-fine framework.

Differential algorithms are also known as variational algorithms. They rely on finding the displacement field by minimising an energy function. This function generally comprises a data and global smoothness term, and minimised in a variational framework. Over the years variational methods attracted increasing attention and became more popular. This popularity may be attributed to their accuracy and the fill-in affect that such algorithms provide [4], [3]. The general energy function of variational optical flow can be written as follows:

$$E = \alpha_1 E_{data}(\mathbf{u}) + \alpha_2 E_{smooth}(\mathbf{u}) \tag{2.1}$$

where $\mathbf{u} = (u, v)$ are the displacements in the two directions x and y, and α_1, α_2 are weights between the data and the smooth term.

Optimisation techniques are an important factor which have an effect on the efficiency of each algorithm, whether in terms of accuracy or computation speed. Euler- Lagrange equations provide an efficient solution, and have been used by many algorithms. The output of the Euler- Lagrange system is a set of equations that can be solved either by direct division, or using a suitable iterative solver such as Successive Over Relaxation (SOR). If the resulting set of equations is non-linear, a fixed point iteration may be utilized to linearise each non-linear term [12], [13]. Several algorithms minimise the optical flow energy function in a dual (primal-dual) formulation. In this case the minimisation is split into two stages, the first stage is a minimisation for the data term and can be a point-wise minimisation step. The second can be considered as a de-noising process [22], [14], [23], [24]. A detailed discussion on the primal-dual algorithms can be found in Chapter-4.

The solution of variational optical flow equations is computationally expensive, and often

cannot be implemented in real time. Some work has been dedicated to reach real or near real-time performance. Bruhn et al. [25], [26] proposed the use of a Bi-directional multi-grid method to speed up the computation of variational optical flow. Multi-grid methods are numerical algorithms used to solve differential equations [27], and offer a fast numerical scheme for solving linear equations. It was applied to the Combined Local Global method [12]. The implementation [25] was able to achieve a speed of 18.229 frames/second on the *'Yosemite without clouds'* sequence [1]. Another multi-grid computation algorithm was proposed in [3], this time the authors implemented the variational algorithm that was proposed by Brox et al.[2]. This algorithm (i.e. [2]) is computationally demanding due to the non-linearised data term and the use of non-quadratic penalisers, in addition to the use of a very fine multi-resolution scheme (Section-3.1.5). The algorithm was able to achieve 6.651 frames/second with good performance in terms of accuracy. Multi-grid methods were also used to calculate 3-D optical flow. Kalmoun et al. [28] implemented an extension of the 2-D Horn-Schunck [1] algorithm. The implementation was used to compute motion of cardiac Computed Tomography (CT) images. Further to this the authors investigated the parallel implementation of this algorithm. The parallelisation was implemented on a cluster of CPUs, and it was found that there is a speed up in the performance as the number of processors increase. Despite the fast computation, the implementation had high memory cost as data volume increases. Multi-grid methods offer a good solution to speed up implementation on CPUs for some low and medium resolution images. However multi-grid methods are known to be problem specific and require very complicated implementation. In addition to that it did not give real-time performance on modern test sequences [29].

The computation of optical flow field via variational techniques requires a lot of calculations at pixel level. Therefore the use of parallel computation seems a plausible way to increase speed of calculation of such algorithms. Several implementations are available in the literature that attempt to parallelise optical flow computation on Graphic Processing Units (GPUs). Some implementations of variants of the local method of Lucas-Kanade [10] algorithms are available in the literature. For example the CUDA implementation of a dense version of the Lucas-Kanade algorithm can be found in [30]. Another implementation can be found in [31], and the authors reported high implementation speed in this paper. Some other examples for Lucas-Kanade implementations are also available such as the high speed VLSI (Very Large Scale Integration) implementation [32]. An example for a Field Programmable Gate Array (FPGA) implementation can be found in [33]. The lack of a smoothness (regularity) term in the Lucas-Kanade algorithm made the computation required in such local methods simpler, and thus it was found suitable for fast implementation.

[1] http://cs.brown.edu/~black/images.html

The performance of such implementations varies depending on the hardware used and the efficiency of the code architecture. In addition to that the choice of the optical flow computational algorithm plays an important role. The global optical flow algorithms are more computationally demanding, the dense solution provided by these algorithms increases the computational effort. Global methods generally produce a dense displacement field and requires intensive computation for each image pixel, unlike the local methods that may give sparse results (see Section-3.5). The work presented in [29] implements the Complementary Optical Flow proposed in [15], they achieved near real-time results. The parallelisation was implemented on an NVidia GPU with CUDA architecture. In addition to GPU parallelisation, this implementation used the *Fast Explicit Diffusion* (FED) method which was proposed by Greweing et al. [34]. FED is a numerical scheme for implementation of diffusion filters, which has been shown to be efficient and accelerates the computation. A fast GPU implementation of the large displacement optical flow [13] was presented in [35]. The authors aimed to implement a fast point tracker that gives dense results, they compared their results to the Kanade-Lucas-Tomasi (KLT) point tracker [36], [37]. The results were not 100% dense due to the removal of several points such as those in areas without structures. However the implementation offered a speed-up of 78 times in comparison to the KLT tracker.

Another group of papers used a dual formulation minimisation with the total variation $TV - L^1$ optical flow, such as the algorithms proposed by Zach et al. [22], and the improved version proposed by Wedel et al. [14], in addition to the Anisotropic Huber-L^1 optical flow [23]. The duality formulation for optical flow offers an easier implementation, additionally the structure of such algorithms enables an easy parallelisation on GPUs. Moreover, the duality provides faster convergence rates [38]. The three aforementioned algorithms in this paragraph had a parallel implementation on a GPU, and it was found to give a good speed. For example the algorithm in [22] was able to estimate the flow field of a video with frame resolution of 320×240 and a frame rate of 30 frames/second.

In addition to the continuous formulation of the general optical flow equation 2.1, it is possible to derive a discrete version of the optical flow energy function. In the discrete case, the optical flow computation can be seen as a problem of assigning a proper label for the displacement [39]. The displacement is found as the best label among a set of labels. Since the displacement is found as the best label among a number of specific labels, the solution lacks sub-pixel accuracy. In fact there is a trade-off between the accuracy of the solution and the time required for the optimisation. The optical flow energy function in the discrete

case can be formulated as follows [39]:

$$E = \sum_{\Omega_D} \left[E_{data} + \sum_{\mathcal{N}} E_{MRF} \right] \tag{2.2}$$

where E_{MRF} is the counterpart of E_{smooth} in the discrete settings and $\mathcal{N}$ denotes the neighbouring pixels. The focus of this thesis is on the continuous variational formulation, but in order to complete the picture of this literature review, a brief discussion on the MRF formulation of optical flow is presented here. An example for such formulation can be found in the paper presented by Sun et al. [11], in this paper the authors used an MRF model for the optical flow energy function. The aim was to learn the statistical model of both parts of this function (data and smoothness terms). A Steerable Random Field (SRF) [40] was then used to model the smoothness term to preserve flow boundaries at image edges. According to [11], the optical flow field can be formulated as follows:

$$p(\mathbf{u},\mathbf{v}|\mathbf{I}_1,\mathbf{I}_2;\Omega) \propto p(\mathbf{I}_2|\mathbf{u},\mathbf{v},\mathbf{I}_1;\Omega_D).p(\mathbf{u},\mathbf{v}|\mathbf{I}_1,;\Omega_S)$$

The first term describes how to obtain the second image $\mathbf{I}_2$ using the displacement fields u,v and the first image $\mathbf{I}_1$, this corresponds to the data term in Equation-2.1. The second term is the prior knowledge of the flow fields, which corresponds to the smoothness term in Equation-2.1. The authors in this paper attempted to learn the statistical properties model of both terms (data and smoothness). Additionally the authors followed [41] in assuming that the horizontal and vertical component of the flow are independent for simplicity. The use of MRF and discrete optimisation methods to find optical flow actually dates back to the early nineties. Vlontzos and Geiger [42] used MRF modelling to find optical flow, their algorithm was based on area matching where they defined a matching window and search window. Heitz and Bouthemy [43] used a Coarse-to-fine framework (Multi-resolution) in their algorithm, they relied on fusing the motion detected from two sources: image gradients and sparse features. The information is fused via a framework based on Bayesian estimation theory and MRF models.

Combinatorial methods are used for optimisation in discrete settings. Among those methods *'Simulated Annealing'* [44] was used for optical flow calculations [45]. However simulated annealing is very computationally demanding and may take a very long time to converge.

Graph cut is another combinatorial optimisation method, it is an efficient and fast way of minimisation of a MRF based energy function [46], [47]. Graph cut is based on the problem of estimating the minimum cut with maximum flow (Min-cut/Max-flow) in a graph con-

sisting of nodes and directed edges [48]. Graph cut algorithms were used in many image processing tasks, such as motion estimation and stereo matching [43].

Recently convolutional Neural Networks (CNNs) were also used to find optical flow [49]. This was motivated by the recent advances that enabled a per-pixel processing [50], [51]. In [49] a CNN was trained using backpropagation to estimate optical flow field, the problem was formulated as a supervised learning task and several architectures were tested. The computation time was relatively short the it was possible to reach a 5 to 10 frames/second, However the neural network layers were implemented only on GPU. One of the problems of such an approach is the need for a huge training data to be available. In the final stage a variational step was applied to refine the estimation accuracy.

Despite the recent advances in several non variational methods for estimating optical flow such as the discrete methods, variational algorithms are still common. Most of the non variational algorithm lack the sub-pixel accuracy and the filling-in effect that variational methods provides. Even if non variational method is used to estimate the optical flow displacement field, a final step of refinement is used in many algorithm to improve the final estimation [52], [49], [53], [54], [55].

2.1 Data Fidelity Term

In the context of finding the displacement field between two images, it is assumed that the brightness of each pixel does not change over time, this can be formulated as follows:

$$I(x,y,t) = I(x+u, y+v, t+1) \tag{2.3}$$

where (x,y) are the indices of pixels in the two dimensional rectangular image sequence I with region domain Ω. In other words the intensity level of a pixel in the first image with location specified by the indices $(x,\ y)$ is equal to the intensity level of the pixel in the second image at location $(x+u,\ y+v)$. Where the first image taken at time t and the second image taken at time $t+1$, and $(u,\ v)$ are the displacements of the pixel in the x and y directions respectively as stated earlier. This equation is linearised using the Taylor expansion to obtain what is known as the optical flow constraint:

$$I_x u + I_y v + I_t = 0 \tag{2.4}$$

where the subscripts denote partial derivatives, specifically I_x and I_y are the derivatives of the image in the x and y directions respectively, and I_t is the time derivative and is actually the difference (subtraction) between the two image.

One of the earliest attempts to deal with optical flow computation was in the algorithm developed by Horn-Schunck [1]. In this marquee work, optical flow was computed globally by minimising an energy function in a variational framework. Similar to the general model in 2.1 the energy function mainly comprised two terms, a data fidelity term and a regularization term. The data fidelity term favours pixels with similar intensity level, while the regularization term imposes a smooth flow field,

$$E_{HS} = \int_{\Omega} [(I_x u + I_y v + I_t)^2 + \alpha(u_x^2 + u_y^2 + v_x^2 + v_y^2)]\, dxdy. \tag{2.5}$$

The first term in the above equation is actually a quadratic version of the optical flow constraint in Equation-2.4. The smoothness term chosen in [1] was the sum of the squared magnitude of the gradient of the displacement field. Contrary to the global solution proposed by Horn-Schunk, local methods assumes that the displacement field are the same for a small local patch, and the optical flow field calculated here may not be dense. The earliest version of this was the Lucas-Kanade [10] algorithm, which estimates optical flow by minimising the quadratic equation:

$$E_{LK} = \mathbf{u}^{\top} J_{\rho}(\nabla_3 I)\mathbf{u}. \tag{2.6}$$

The Lucas-Kanade method uses a rectangular Gaussian kernel for flow integration, with a standard deviation ρ often referred to as the integration scale [12]. The size of the window should be big enough to estimate optical flow. A small window may not be sufficient for finding the displacement field due to the aperture problem [1]. On the other hand, the use of a bigger window results in less accuracy and more possibility of blurring across objects and motion boundaries. To preserve edges in such methods, a possible way is to use a window with varying size such as the algorithm in [56], where windows are changed for each pixel depending on local variation of intensities around that pixel. A different approach is the use of image segmentation. Ren in [57] used local grouping to decide the motion of pixels near edges. In this paper boundaries including corners and edges are detected then pairwise affinities between neighbouring pixels are computed using intervening contours [58]. This is done near boundaries only, to speed up the operation.

It is known that the global methods give a dense flow field while the local methods are more

robust to noise. Bruhn et al. [12] combined both local and global method to improve the performance and to obtain a flow field that is both dense and more robust compared to other global methods. In order to compute the optical flow field, the authors proposed to minimise the following energy function:

$$E = \int_{\Omega} (\mathbf{u}^{\mathsf{T}} J_{\rho}(\nabla_3 I)\mathbf{u} + \alpha|\nabla \mathbf{u}|^2) d\mathrm{x} \tag{2.7}$$

where $J_{\rho}(\nabla_3 I) = K_{\rho} * (\nabla_3 I \nabla_3 I^{\mathsf{T}})$ and K is a Gaussian kernel with standard deviation ρ.

As a result of smoothing in the latter equation, the value of intensity level of a pixel is replaced with a weighted sum of pixels' intensities in the neighbourhood, which increases the robustness. However, the smoothness also will have the same effect across boundaries, and the motion boundaries become blurry. To mitigate this effect Drulea et al. [4] replaced the Gaussian smoothness kernel with a bilateral filter [59] [60]. In the Gaussian kernel case the weight of pixel contribution decreases as the pixel is further from the centre, the bilateral filter on the other hand decreases the weight also as the pixel intensity value has a bigger difference from the central pixel. It is expected that pixels across boundaries have difference in intensities values, thus a bilateral filter suppresses the smoothing effect across objects boundaries in the image.

The data fidelity term is an important element of the general optical flow energy function 2.1, it favours pixels with similar intensity (or colour) levels. As pointed out earlier, the data term is linearised resulting in the well known Optical Flow Constraint *ofc* [1], [12], [4]. A Non-linearised version of the data fidelity term was also considered in [2], [61], all the minimisation is done in the C2F framework. The basic assumption for the calculation of optical flow is that the illumination of a certain point does not change over time, any changes in the illumination may lead to error in calculations. Therefore it is useful to make the data term more robust and resistant to such illumination variations.

Illumination variation in images can be attributed to several reasons, such as change of illumination in the scene between the two images (locally or globally), sensor noise, reflections and shadows [14], [62]. Wedel et al. [14] proposed an improved algorithm for the primal-dual algorithm in [22] to increase the robustness to illumination changes in the images. The authors proposed to decompose the images into structure and texture parts. The structure part outlines the big objects in the scene, while the texture one contains the fine scale details. It is expected that the shadows and shading reflections show up on the structural part while the textural part is noisier due to sensor noise and sampling artefacts. The structural part is found using total variation minimisation based on Rudin, Osher and Fatemi [63]. The tex-

tural part is computed as the difference between the original image and the structure image. The input to the optical flow algorithm a mixed version of the structural and textural parts.

Changes in illumination can be considered as outliers, similar to discontinuousness in the smoothness term, it is desired to penalise such changes less severely. Black and Anandan [64] [65] [66] considered using robust statistics for computing optical flow, a technique proved its effectiveness as many subsequent papers used such estimators [12], [2], [13], [67]. The illumination variation here is considered as outliers, therefore this method will not be very useful in the case of large illumination variation that affect large areas.

Augmenting the data term with the image gradients constancy was found also to be useful, it is also assumed that the image gradients do not change over time:

$$\nabla I(x,y,t) = \nabla I(x+u, y+v, t+1) \tag{2.8}$$

Adding such constraint to the data term can be useful especially in the case of translational displacements [2]. This term means that even if the illumination changes in the two images it is still relatively constant in relation to the surrounding points. The data fidelity term will be composed of the intensity and gradient differences:

$$E_{data} = E_{intensity} + \gamma\, E_{gradients} \tag{2.9}$$

where γ is a weight factor. Several algorithms followed this method and used the gradients constraint to improve the robustness to illumination [2],[13], [61], [67].

Xu et al. [68] found that the combination of intensity (or colour) values and the gradient does not always contribute optimally to the optimisation problem. In fact sometimes the intensity values can give better estimation, while in other cases using the gradient values alone can give good results. The authors of this paper argue that using two constraints in the data term can be less efficient in modelling pixel correspondences than only using one of them. They showed empirically that the data cost for pixels with regard to different displacements are not always minimum using both data constraints. Therefore they proposed to switch between the two constraints by introducing a binary weight map. However this complicates the minimisation, therefore Mean Field approximation [69] was used to simplify the problem.

Adding the gradient constancy to the data term can help to increase robustness against what is known as *'Additive illumination changes'*. However, real-world scenarios may include more complex illumination changes to deal with [67]. In case of more complex illumination

variations, it may be useful to model these changes and assume that the illumination changes can be decomposed into additive and multiplicative components. One of the attempts to model these changes was by Negahdaripour [70], where he proposed the following model:

$$I(x,y,t) = M(x+u,y+v,t+1)I(x+u,y+v,t+1) + C(x+u,y+v,t+1) \tag{2.10}$$

where M, C are the multiplicative and additive factors respectively. Other models of illumination changes also available [62].

So far the discussion has been limited to grey scale images, that depend only on intensity levels. Colour images on the other hand may have an advantage (in terms of illumination robustness) over grey scale images in that they contain more photometric information [71]. This means that it is easier to derive photometric invariant models. As the use of colour spaces started to gain popularity over the years, many algorithms adopted the use of colour images to estimate optical flow [72], [73], [71], [74], [75], [67]. Mileva et al. [72] incorporated the non-linearised variational algorithm proposed by Brox et al. [2] into colour space in a multichannel framework to make use of its photometric invariants properties. Several colour spaces were tested (RGB, HSV, and spherical). The RGB colour space describes colours as an additive value of three colours or channels (Red, Green and Blue). Similar to the constraint normalisation [76], normalised RGB spaces were also used [74] [77]. HSV space on the other hand describes colour using three parameters: Hue, Saturation and Value. The Hue represents the colour value, Saturation represents the degree of achromatic/grey component, and the Value is the brightness. The use of HSV colour space to calculate optical flow was shown to be useful for illumination robustness. As was pointed out in [67] [72], the Hue value is invariant to local and global multiplicative illumination changes in addition to local additive changes. These changes are mainly attributed to shadows, shading, highlights and specularities. The Saturation value is invariant to global multiplicative illumination changes mainly attributed to shadows and shading. The Value is not invariant at all.

Steinbrücker et al. [78] compared different types of data terms namely: Point-wise L1, truncated L1, and L1 computed over a patch (CLG case), and proposed a fourth one (Normalized Cross Correlation NCC) which was computed over a local patch and is expected to be more robust to multiplicative illumination changes. The experiments were conducted using the algorithm in [79]. Constraint normalisation was also used in optical flow calculation [67], [76] to amend the affect of the difference in gradients between regions of high and low gradients. Regions of high gradients will dominate the calculation in the locations close to

it [76]. The normalisation helps to reduce stronger enforcement at high gradient locations, as these locations may be caused by noise or occlusion. It was shown in [15] that the use of normalisation improves the performance. The normalisation term as was proposed in [15] can be formulated as:

$$\mathscr{N} = \frac{1}{|\nabla I|^2 + \zeta^2}$$

where ζ is a small constant added to avoid dividing by zero.

The linearisation in 2.4, although useful for making the minimisation problem convex, introduces a different problem, where it is only possible to find displacement that does not exceed a few pixels. In reality the displacements for pixels can be large, therefore it is desired to find the flow field even for large displacements. To this end, the minimisation is performed in a Coarse-to-Fine (C2F) framework (also called Multi-resolution, Multi-scale), where the displacement is computed in a coarsened version of the images to guarantee small displacements. This computed optical field is then propagated to a higher resolution image via interpolation. Some papers used a non-linearised version for the data term, this can be found in [2], [80], [61], but these algorithms still worked in a C2F framework.

Brox et al. in [2] opted to use the non-linearised version of the optical flow constraint 2.3, this enabled them to detect more complex motion (e.g. rotational) so the algorithm becomes rotationally-invariant, however the non-linearity introduced in the energy function makes the optimization non-trivial as it easily gets trapped in an unwanted local minima, therefore the authors opted back to linearisation at the numerical solution stage. The data term was formulated as follows:

$$E_{data} = \int_{\Omega} |I_2(\mathtt{x}+\mathtt{u}) - I_1(\mathtt{x})|^2 d\mathtt{x} \tag{2.11}$$

The work in [61] is an extension for the latter paper, various data terms were tested, such as image gradients, Hessian, and Laplacian. The authors found an increase in robustness to illumination changes when using data terms that incorporate image intensities and image gradients. The algorithm in [80] [81] is an earlier attempt to use a non-linearised model in computing optical flow, in fact [80] presents a modification for the algorithm presented by Nagel & Enkelemann in [82]. In these papers the non-linearised version of the optical flow constraint is used, however this energy function is non-convex (as was pointed out earlier), and thus the use of a C2F framework is inevitable. In the C2F framework, the images are coarsened several times and two pyramids of images are created including all the coarsened

versions. The displacement field is then computed on the coarser level to ensure a small displacement, then this flow field is propagated to the next finer level. The propagated displacement field is used as an initialization to compute the displacement in the next finer level. In addition to that the displacement field is used to warp one image towards the other, so the optical flow field is computed between one image and a warped version of the other image. The C2F is not without its problems, for example when the image is coarsened, one would expect the small details in the original images to be lost, therefore it would not be possible to find the flow field for these small structures when their displacements are large.

On the other hand, descriptors matching has been relatively successful for many applications such as object detection [83], and structure from motion SfM. Due to the nature of the descriptors and its uniqueness it does not have limitations in the case of large displacements. For this reason Brox et al. in [13], [84] integrated sparse descriptor matching into a variational framework with the aim of enhancing the performance of optical flow calculation in the presence of small structures. In that paper the optical flow was calculated by minimising an energy function comprising data and global smoothness terms, in addition to that a descriptors matching term was added as a soft constraint to the energy function. The purpose of this term is to prevent the solution of the optical flow drifting away from the correct flow field. Several types of descriptors were tested in [13], such as the Histogram of Oriented Gradients (HOG) [83], and Scale-Invariant Feature Transform (SIFT) [85].

Li Xu et al. in [68], [86] showed that the problem with small scale structures occur also in the case of small displacements, basically when small motion structures exist near a large scale motion structure. In this paper the authors proposed a modified C2F algorithm, and called it the Extended Coarse-to-fine framework (EC2F), where the initialization of the displacements is chosen from several motion candidates obtained from the coarser layer displacement flow, sparse descriptors (SIFT) and patch matching. A labelling process is then adopted to choose the optimal flow candidate. The authors used Quadratic Pseudo-Boolean Optimization (QPBO) [87], the algorithm also included an occlusion awareness refinement stage to enhance the result in the presence of discontinuities. Although this algorithm gave very good results, it is computationally expensive as it includes feature detection and matching and labelling optimisation on each pyramid level in addition to the variational continuous optimisation.

Patch matching was also used as a correspondences between images to estimate optical flow. Chen et al. in [53] found that matched patches from Nearest Neighbour Fields (NNF) [88] contain a percentage of approximate motion fields. They proposed to find Optical flow through finding the patch matches, obtaining several motion candidates which accounts for

translational plus rotational motion patterns via Random Sample Consensus (RANSAC) on image patches in addition to perturbation on motion patterns. The final motion candidates are chosen through a labelling procedure using graph cut [47] and QPBO fusion, and the problem of optical flow is formulated as a motion segmentation problem. Finally, the obtained motion field is used as an initialization for a variational optical flow computation to obtain sub-pixel accuracy. The fact that the initial motion field obtained by patch matching contains a high percentage of the motion field makes it very good for finding optical flow for small scale objects, where the C2F is not used. However the problem with using NNF in this algorithm is the ambiguity introduced due to similarity in patches, especially in texture-less areas. The size of the patch is also important and was pointed out in the paper. A more recent paper presented Edge-preserving large displacement Optical flow [89]. The authors used bilateral weights to the patch matching cost to preserve discontinuousness. To speed-up the algorithm, the images are down-sampled twice by a factor of 0.5.

In [55] sparse correspondence were used in rather a different way to obtain quasi-dense feature correspondence and work in a bottom-up fashion in the image hierarchy (i.e. from finest to coarsest layer) to obtain a response map which will be used later in a C2F variational solution approach. Revaud et al. [54] used dense correspondences to estimate optical flow by sparse-to-dense interpolation that respects edges. The results of this interpolation is used as an initialisation for a variational optical flow estimation. Leordeanu et al. [90] tried to eliminate the use of C2F and proposed to expand sparse features to approximate a dense motion field using a locally affine model, and the final motion field is obtained using continuous total variation method. Steinbrücker et al [79] proposed a new framework for calculating optical flow without a C2F framework. It is related to the duality approach presented in [22], but changed the second step to be a complete search of pixels. Obviously this means that this algorithm is computationally expensive.

2.2 Smoothness (Regularisation Term)

The smoothness term in equation 2.1 assumes that the displacement field to be calculated is smooth across the image. When computing optical flow certain assumptions are made, one of these assumptions is the spatial constancy, which means that neighbouring pixels in an image move together in space. Hence a regularisation term is added to the data term to penalise diverse displacements of neighbouring pixels. Of course this is not always the case because this assumption is violated at image motion boundaries and objects occlusions. These violations can be treated as outliers. One choice for a smoothness term is the quadratic

penaliser $\Psi(s) = s^2$ used by Horn-Schunk [1], the smoothness was chosen to be the sum of the squared flow gradients:

$$E_{smooth}(\mathbf{u}) = |\nabla u|^2 + |\nabla v|^2 = (u_x^2 + u_y^2 + v_x^2 + v_y^2). \tag{2.12}$$

The quadratic smoothness used here penalises the flow field (and hence outliers) severely in all directions, making it sensitive to such outliers. As a consequence the motion boundaries are over-smoothed. This creates undesired blur across motion boundaries.

It seems logical to replace this regularity with one that permits piecewise smoothness, such as the first order norm total variation $TV - L^1$ regularier, which is an isotropic rugulariser (i.e. penalises the flow field in all directions equally). This regulariser is more robust and efficient in handling discontinuousness across boundaries. However the problem with such regularisation is that they are not continuously differentiable. Chambolle in [91] presented a numerical scheme to solve $TV - L^1$ minimisation with applications to image denoising and zooming. This algorithm was then used by Zack et al. [22] to compute optical flow in the spirit of primal-dual optimisation. In that paper the minimisation problem was broken down into two alternating steps. The first step is minimised by keeping an auxiliary variable fixed and solving a total variation-like energy function. The second step is a point-wise thresholding minimisation depending on how close the auxiliary variable is to the actual solution. The minimisation works under the Coarse-to-Fine framework to ensure the computation of large displacements.

Several algorithms modified the $TV - L^1$ ([4], [23], [92]) to enhance its edge-preserving properties. This is done by adding a strictly positive decreasing function to the smoothness term [93] and thus reducing the smoothness across image edges and corners. A regulariser was proposed in [4] of the following form:

$$E_{smooth} = \int |D.\mathbf{u}| = \int (|D.u| + |D.v|) \tag{2.13}$$

where D is the diffusion coefficient, which was chosen in [4] to be:

$$D = e^{-\alpha |\nabla I|^\beta} \tag{2.14}$$

where α and β are constants. In addition to that the authors in [4] used Bilateral filtering in the data term to suppress propagation (as discussed in Section-2.1 in more detail). It was proposed in [11] to add another regularisation term to favour rigid body motion, as this type

of motion is the only one available in images with moving camera and rigid scenes, and the common motion of images with moving rigid bodies.

Since the regularisation term in Equation-2.13 is adaptive based on the image gradients, this smoothness term can be denoted as an image-driven regularity term. The use of such regularisation will lead to over-segmentation in the computed flow field especially in areas with high textures. This is due to the fact that image boundaries do not always coincide with motion boundaries. In fact motion boundaries can be seen as a subset of image boundaries [94].

The $TV-L^1$ regularisation favours a piecewise constant solution. This will become clear in untextured areas where the computed flow field suffers from what is known as the *'stair-casing effect'*. The stair-casing phenomenon is caused by the use of a piecewise constant smoothness term such as the total variation L^1 norm [23], [95]. Which can be seen as artificial boundaries in the computed flow field [96], obviously this can decrease the accuracy of the flow field.

To deal with this Werlberger et al. [23] proposed an anisotropic, image-driven, Huber regularisation term instead of the isotropic $TV-L^1$ in the smoothness term. The Huber-L^1 norm penalises outliers less severely, while the addition of a symmetric, positive definite diffusion tensor impedes the propagation across image boundaries. The minimisation was performed in a dual formulation and using an auxiliary variable similar to the previous algorithm proposed by Zach et al. [22]. Other papers to deal with the stair-casing effect can be found in the literature. For example Trobin et al. [97] opted to use higher order derivatives in the smoothness term to penalise only deviation from an affine function. This is based on the fact that the second order derivative of an affine flow function is zero, hence no weight is assigned at this point. In [95] the authors presented a generalised notion of $TV-L^1$ to an arbitrary order of derivatives which does not lead to stair-casing effect. This notion was also used in optical flow algorithms, an example is the work done by Braux-Zin et al. [98]. In this paper the author used the 'Total Generalized Variation regularization' and used AD-Census in the data fidelity term [99].

As pointed out earlier, in order to improve the performance of optical flow algorithms at motion boundaries, it is a good idea to replace the quadratic regularization with a non-quadratic one that permits piecewise smoothness. This is due to the nature of the optical flow field which is piecewise smooth. Beside using the L^1 norm, several algorithms tried to approximate the work of the L^1 norm behaviour, these penalisers are borrowed from Robust Statistics [64] and applied in the field of optical flow and also image diffusion. Weickert

in [100] replaced the smoothness term in [1] with the non-quadratic term in the form:

$$\Psi(\mathbf{s}) = \varepsilon^2 \sqrt{1 + \frac{\mathbf{s}^2}{\varepsilon^2}} \tag{2.15}$$

where $(\varepsilon^2 > 0)$. Which is similar to the robust estimator proposed by Charbonnier et al. [101] and was used by Bruhn et al. [12] in their CLG optical flow. Another example is given by Black-Anandan in [64] [65] [66], inspired by robust statistics. They proposed the use of Lorentzian estimator :

$$\Psi(\mathbf{s}) = \log(1 + \frac{1}{2}(\frac{\mathbf{s}}{\varepsilon})^2). \tag{2.16}$$

The proposed estimator penalises outliers less severely in the computation of flow field and hence improves the performance on the image motion boundaries. Later on many algorithms used the robust penaliser in the form $\Psi(s^2) = \sqrt{s^2 + \varepsilon}$ [2], [61]. Several other papers using non-quadratic functions can also be referred to here, such as [102], [61].

In the context of preserving edges in optical flow computation, smoothness can be classified into image-driven and flow-driven terms. A good discussion in the literature on the classification of the smoothness term can be found in [103], also in [104] a good taxonomy was provided and the connection of a smoothness term with diffusion filters was shown. The edge-preserving smoothness term was classified in [104] into three types, isotropic image-driven, anisotropic image-driven and isotropic flow-driven. In addition to that an anisotropic flow-driven smoothness was proposed.

An image-driven smoothness works by reducing the penalisation across image discontinuities. An example for isotropic image-driven smoothness can be found in [81]. In this paper the authors modified the Horn-Schunck smoothness term by multiplying it with a decreasing strictly positive function of the image gradients:

$$E_{smooth}(\mathbf{u}, \nabla I) = g(|\nabla I|^2)(|\nabla \mathbf{u}|^2) \tag{2.17}$$

Hence, the smoothness effect decreases across image discontinuities as these regions usually have high gradients, and increases in smoother image areas. The notion of isotropy here stems from the fact that the function g is directionally-independent. In other words it does not depend on the direction of the image gradient, and penalisation takes place in all directions equally. As a consequence for being 'isotropic' the fill-in effect along image (and motion) edges will also be affected. Contrary to this is the anisotropic image-driven smooth-

ness which is directionally-dependant. Early examples for this smoothness can be found in [80], [105], where they used a diffusion tensor instead of the directionally-independent function g in equation 2.17. The smoothness term used in [80] can be written as:

$$E_{smooth} = \int trace(\ (\nabla \mathbf{u})^{\mathsf{T}} D(\nabla I)(\nabla \mathbf{u})\ ) \tag{2.18}$$

Here $D(\nabla I)$ is a regularized projection matrix perpendicular to ∇I:

$$D(\nabla I) = \frac{1}{|\nabla I|^2 + 2\lambda^2}\{\nabla I \nabla I^{\mathsf{T}} + \lambda^2 \mathbb{I}\} \tag{2.19}$$

where $\mathbb{I}$ is the identity matrix.

The problem with image-driven smoothness as was discussed earlier is that it changes the effect of smoothness in relation to the image gradients. This may lead to over-segmentation in the estimated displacement field. Hence, it is more suitable to design the smoothness term to be adaptive with the flow field itself. The quadratic regulariser used by Horn-Schunck is not suitable for such purpose because of its strong penalisation, including areas across motion boundaries. Therefore it is inevitable to refrain from it and use a piecewise smooth function, in fact the smoothness terms discussed earlier (e.g. [100], [64], [65], [66], [101]) are considered isotropic flow-driven smoothness terms.

Xiao et al. [106] used bilateral filtering [59] in the smoothness term to deal with occlusion. As was pointed out in this paper, theoretically the occluded regions should not have a displacement vector, however in practice they do have. This displacement is induced from the smoothness term and the fill-in effect. The authors in this paper used this type of filter to harness the smoothing across motion edges. The optimisation was decoupled into two steps (as the two terms work in different domains). One optimisation step updates the displacement field and the second minimises the data energy, and a bilateral filter was applied in the smoothness term.

Another type of smoothness was used which depends on information contained in the image and motion fields. The combination of this may give better estimation for the motion edges. Thus far in this section, the main type of smoothness discussed are isotropic and anisotropic image-driven smoothness, in addition to isotropic flow-driven smoothness. Ideally the desired smoothness term would be one that depends on the motion discontinuities rather than image discontinuities. This is because image-driven smoothness may introduce over-segmentation in the estimated flow field. In addition to that the smoothness term would be directionally-dependant (i.e. anisotropic), such as to harness propagation across motion

discontinuities while encouraging this along them. In [15], [67], [11], [40] the authors used what can be described as a hybrid smoothness term, they made use of the structure tensor and its eigensystem. The structure tensor is a 2×2 matrix that contains information about the structure orientation of a certain location in the image. It is possible to extract two orthonormal eigenvectors for such a matrix one of these vectors pointing across image structures while the other points along them. In [15], [67] a smoothness term was proposed which has a form similar to the following:

$$E_{smooth} = \Psi\big((e^T \nabla u)^2\big) + \Psi\big((e^T \nabla v)^2\big) \tag{2.20}$$

where $e \in \{e1, e2\}$ are the eigenvectors of the structure tensor matrix. ∇u is the gradient of the displacement in the x direction (i.e. $\nabla u = (u_x, u_y)$), and ∇v is similar but for the displacement in the y direction. The function Ψ is a robust function penaliser (e.g. Equation-2.16). In this way the smoothness term magnitude will be proportional to the motion field, while the direction is dictated by the structure tensor eigenvectors. Hence the problem of over-segmentation is avoided. This idea was originally adopted from [11], [40] which was proposed to work in discrete settings, and was adapted to work in continuous settings in [15], [67].

A closely related notion to the structure tensor, is a representation called *'Motion tensor'*, also denoted sometimes as *'Orientation tensor'*. The motion tensor is a 3×3 symmetric positive semi-definite matrix, which contains information about the local orientation. The main difference from the structure tensor, is that it extends it to obtain spatio-temporal derivatives. A Motion tensor was used in [107], [108] to compute the velocity in an image sequence. The velocity (motion) was assumed to be constant over a certain region in a similar assumption to the local optical flow of Lucas-Kanade [10]. Zimmer et al. [67] used the spatio-temporal representation to extend their algorithm in the time domain to obtain a temporally smooth flow field.

So far in this section, the main smoothness terms discussed were spatially-dependant. Usually optical flow is required to be computed for a sequence of images (e.g. video). Hence it would be plausible to think of using a smoothness term in the time domain. Black and Anandan assumed in their paper [65] that the acceleration of pixel patches in an image plane over time is constant, and acceleration changes slightly due to noise. Hence it is possible to estimate the displacement in the next frame. The authors used this in addition to spatial coherence in this paper. Weickert et al. [103] used spatio-temporal smoothness regularisers to extend flow-driven spatial smoothness. The flow-driven smoothness was obtained using a

non-quadratic robust function. The authors in this paper used a function of the form shown in Equation-2.15. The spatio-temporal smoothness proposed took the following form :

$$\Psi(\mathbf{u}) = \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2) \tag{2.21}$$

where ∇_3 denotes a spatio-temporal nabla operator $(\partial_x, \partial_y, \partial_t)$. The authors also reported a 50% increase in the computation time compared with the spatial case.

The smoothness terms discussed earlier are based on the assumption that neighbouring pixels have similar displacements, in other words the displacement is assumed to be locally similar for those pixels. Non-local smoothness augments this assumption to include a number of pixels in a certain neighbourhood. Several algorithms used non-local smoothness terms to extend the range of different types of the previously discussed regularity terms. A non-local version of the total variation regularisation was presented in [38]. The smoothness term included (in addition to $TV - L^1$) a term which is a function of both colour similarity and distance proximity. The penalisation of such a smoothness term may resemble to an extent that of the bi-lateral filter (see Section-3.2.2). In this way, and by including a bigger neighbourhood in the calculation, the robustness increases. In addition to that and because the term is adaptive and gives more weight to pixels with similar colours, the edge-preservation performance is improved. This can be attributed to the fact that pixels on the sides of boundaries usually have a big colour difference compared to pixels on the same side.

Krähenbühl and Koltun [109] extended the general optical flow equation (Equation-2.1), and added a non-local term. The proposed energy function will take the form:

$$E = E_{data} + E_{smooth} + E_{non-local}$$

where the term $E_{non-local}$ refers to the non-local smoothness term, and E_{smooth} is the traditional regularity term which can be any of the previously discussed smoothness (local) terms. The problem with non-local regularisers is their computational complexity which tends to increase quadratically with the size of the neighbouring window [109], although this was improved to a certain extent in the last paper via approximation using a mixture of exponentials, a concept related to Gaussian Scale Mixture (GSM) [11].

2.3 Optical Flow Applications

Optical flow cue is a very important tool which has wide use in image processing and computer vision. It has been used in many areas across those fields such as robot navigation, medical imaging, Structure-from-Motion and 3D composition. This section aims at providing a brief literature review for the main optical flow applications, and tries to show its importance in the image processing field. Living creatures travel through their environment based on information collected using their senses, of which vision provides the richest information. In autonomous vehicles this corresponds to using sensors to do this job. Several types of sensors may be used for this purpose such as sonar, laser and infra-red or even a mix of several types of sensors [110], [111], [112]. However visual navigation has been attracting a great deal of research over the years. The choice of vision as the main information source for navigation may seem obvious due to the rich information it provides. In addition to that, researchers are trying to mimic the navigation systems that living creatures use to traverse environments [113].

Perhaps a fundamental operation in robot and autonomous navigation is the ability to avoid obstacles existing on the path to be traversed. In order to do that, obstacle detection is needed. Several methods have been used for this purpose, among those are the ones relying on previous knowledge of the environment, called *'map-based navigation'*. If such information exists then it is easy to design suitable algorithms to traverse the environment, the maps can vary in the details they contain. Once the map is provided the robot can find its location *('Localize')* in relation to the obstacles in the environment. An example of map-based navigation is the *'Occupancy grid'* where obstacles are represented as a 2D projection of the volume [114], [112].

Santos-Victor et al. [115] proposed a biologically inspired robot navigation system. The idea was inspired by the navigation of honeybees and insects. In this system, images from two cameras where used to find the optical flow field. These two cameras were mounted on a mobile platform pointing horizontally in opposite directions. The control of the robot was of a reflex-like control in relation to the computed optical flow. The difference of the average flow field computed by the two cameras was used to control the trajectory of the mobile robot. The aim was to balance the flow computed by the opposite cameras which were directed to the walls, a main requirement was that the walls should be textured. The flow field was computed using the optical flow constraint (Equation-2.4). Since the camera is fixed on the robot, and the robot moves horizontally parallel to the floor, the cameras witness a horizontal displacements flow field only, therefore the vertical component of the

flow field v is set to zero, yielding the following equation:

$$u = -\frac{I_t}{I_x} \tag{2.22}$$

Experiments were conducted in different scenarios to test the effectiveness of the algorithm, wall following and obstacle avoidance, in addition to navigating in a funnelled corridor. This type of visually guided robot navigation belong to the category of mapless navigation, where no prior information about the geometry of the environment is required [112].

Srinivasan et al. [113] discussed several methods for mobile robot navigation, and showed that the observations of travelling insects led to several algorithms for autonomous navigation that rely on motion estimation. In this paper the authors outlined several strategies inspired by insect navigation. The first strategy was range finding which was identified in some insects and animals, which allowed the design of its counterparts algorithms in autonomous movements. For example range finding by a camera moving perpendicular to its optical axis [116]. More detailed information can be obtained by finding the range and orientation of surfaces. It is possible to obtain information by investigating image velocity, for example if the surface is directed uniform to the camera optical axis, image velocity will be uniform. The second strategy is corridor traversing, which is inspired by bees behaviour while flying through corridors, where bees tend to balance the velocity of the walls on the sides. The third strategy discussed in [113] is navigation in a cluttered environment, again inspired from bees behaviour [117], it was possible to design an algorithm to travel in such an environment using image velocity. If a mobile robot is moving in a straight line, the image velocity looks faster in a near area in comparison with an object lying further away. The authors of [113] also discuss the development of two visual odometry systems based on several experiments on honeybees and insects. The first system was proposed to compute the distance travelled using image motion, however the distance calculated is not the actual distance travelled. In order for this information to be useful the robot has to travel in a pre-known environment. Alternatively it is possible to estimate the distance travelled in terms of the time integral of the reciprocals of the image velocity.

In a different approach Camus et al. [118] used optical flow for real-time obstacle avoidance via estimating Time-to-Contact (TTC). In that system images are taken using an un-calibrated camera with a 115 degree field of view. Optical flow was calculated in a correlation-based algorithm [119]. The displacement is searched for in a $(2\eta+1)\times(2\eta+1)$ patch, where η is a parameter value that depends on the expected maximum displacement. The images taken by the camera mounted on the robot have a resolution of 256×512,

which is then sub-sampled to a resolution of 32×64. This sub-sampling helps to speed up the computation of the algorithm, however some image details will certainly be lost. The calculation of TTC involves computation for the velocity at which the object and the robot are approaching each other in relative movement, and the calculation is done in the 2D domain of the image. The x, y components of the optical flow (u, v) can be written as:

$$u = (1/z)(-T_x + xT_z) + (xy\omega_x - (1+x^2)\omega_y + y\omega_z) \tag{2.23a}$$

$$v = (1/z)(-T_y + yT_z) + ((1+y^2)\omega_x - xy\omega_y - x\omega_z) \tag{2.23b}$$

where z is the depth of the point in the environment, and (T_x, T_y, T_z), $(\omega_x, \omega_y, \omega_z)$ are the translational and rotational components of the relative velocity in the environment. It then can be shown that [118]:

$$\nabla(u, v) = \frac{2T_z}{z} \tag{2.24}$$

which shows that TTC depends on the translational velocity in the z direction. The aim of the robot navigation system discussed in this paper is to navigate to a certain point, while avoiding at the same time collision with obstacles on the way. The steering policy depends on what the authors call a *'hazard map'*, which is derived from the flow divergence, and includes obstacles indication and the desired goal direction.

Nelson et al. [120] used flow divergence to estimate (TTC) and hence obstacle avoidance. Their algorithm made use of the motion components projected in the 2D image domain to estimate the motion taking place in the 3D environment. This paper discussed the relation between the different motion types and directional divergence. Relative motion between a camera and a certain point in the surrounding environment can be seen as a composition of three components, namely motion resulting from camera rotation, and others that result from perpendicular and parallel movements of the camera in relation to the point. The authors showed that it is possible to approximate (TTC) using the flow divergence of the computed flow field, and that the rotational component of the motion is not needed to estimate (TTC). It was also shown that divergence at certain points is positive as the camera approaches (relatively) the object points. The authors used one camera mounted on a robot arm for their experiments.

TTC was used along with what is called Focus-of-Expansion (FOE) in robot navigation. Souhila et al. [121] derived those two parameters from optical flow and used them to steer

a robot away from obstacles, and used this to calculate (FOE), which is the point in an image plane where motion everywhere is directed away from. In this sense the point would lie in the centre of the image plane. The aim of the navigation algorithm is to balance the horizontal flow on both sides of the (FOE), thus the control algorithm can be seen as a function of the calculated optical flow. In this paper the authors used optical flow extracted from a sequence of images using the Horn-Schunck [1] algorithm. Low et al. [122] used optical flow as a cue to estimate what they call a *range map* derived from the calculation of (TTC). However, the optical flow calculated was not dense, instead the sparse features were used. The authors used Harris corner detection [123], and the matching was done using normalised cross correlation. (TTC) was then derived for all the detected corners. The range can be easily found using the robot speed information and this operation would result in a map of distances from the camera to each detected corner. Experiments were conducted using a wheelchair type robot in straight and slowly curved paths. Ohnishi and Imiya [124] proposed to use optical flow for dominant plane detection, and used this plane for robot navigation. The dominant plane can be defined as the largest planar area of the image, and in the case of robot navigation it is assumed to be the ground plane in front of the camera where the robot will move. Any part of the image that belongs to the dominant plane can be considered safe to navigate on, while parts that do not belong to this plane can be seen as obstacles and the robot has to avoid colliding into it. The authors assumed that the dominant plane should occupy over half of the image in the initial frame which helped in the detection process. Optical flow was computed using the Lucas-Kanade algorithm [10], and uses this flow to calculate what they call planar flow, which is the displacement field of the dominant plane area, and is estimated via an affine transformation.

Optical flow was also used to remove undesirable motion in videos and images such as shakes and jiggles that exist for example in hand-held cameras. It is used as part of Digital Stabilisation Systems (DIS) which can be found in consumer cameras and mobile telephones [125]. Chang et al. [126] proposed a video stabilisation system that uses cues from optical flow algorithms. The system in this paper calculates the motion field between each consecutive image using the Horn-Schunck algorithm [1], then the rotational and translational global motions are estimated via least square estimation. The calculated motion is then used to counter the undesired shakiness of the video. Liu et al. [127] used optical flow to find motion vectors at each pixel. They denoted this as a *'pixel-profile'*, and used it for video stabilisation by smoothing these vectors. They denoted the optical flow algorithm as *'SteadyFlow'* which is similar to the algorithm in [128], which is proceeded by a global homography transformation to align the two frames, and also includes a discontinuities realisation stage.

Optical flow is also used in image registration algorithms. To register two images is to align them together, this requires finding the correspondences between images which can either be in the form of sparse features or dense displacement vectors. A good description for the relation between image registration and optical flow can be found in the work presented by Lefébure and Cohen [129]. Image registration based on the computed optical flow are sometimes referred to as intensity-based image registration. This type of image registration is usually used in medical imaging, due to its ability to register images with contents that have undergone deformable changes. A good review on deformable image registration can be found in [7]. As pointed out in chapter-1 image registration can be divided into two types, the first type is called multi-modal registration where images to be registered are acquired from different view points may be using different sensors. The second type is called mono-modal where images are acquired from the same view point at different points in time. In this specific case the image registration and optical flow become similar. However, optical flow was used as a part of an algorithm to find multi-modal image registration in [130], where the displacement field was used to estimate complex deformation after initial parametric rigid registration using a particle filter. Optical flow was used also in a similar manner in [131], where an initial estimate for the registration is found, and later an optical flow algorithm was used to refine the registration.

Several optical flow methods have been used for image registration. Pock et al. [9] proposed the use of $TV-L^1$ optical flow for image registration, which follows the algorithm presented in [22] for optical flow calculation. The algorithm used a non-quadratic smoothness term which allowed it to deal with discontinuities and outliers. Experiments were conducted on both synthetic and real clinical images. Keeling et al. [132] used variational optical flow for rigid medical image registration, they penalised the deviation from the optical flow solution.

Optical flow was also used to find Structure-from-Motion (SfM) and monocular depth estimation. Newcombe and Davison [133] used optical flow correspondences as part of a system for scene reconstruction in real-time. In this system the requirement of a real-time performing optical flow algorithm favoured the use of primal-dual optical flow, this is due to its high implementation speed on a GPU hardware [14]. This enabled them to obtain a dense map for the scenes. Optical flow was also used for dense tracking and mapping in real time in several algorithms such as the work of Newcombe et al. [134]. The optical flow version used there is the algorithm proposed in [79], which does not use C2F to estimate the displacement fields and thus does not suffer from losing many details in the images. Another example of dense motion tracking can be found in the work of Sundaram et al. [35], which relies on a real time implementation of the large displacement optical flow by Brox et al. [13].

This section did not aim to give an extensive survey for optical flow application, but rather to give an idea on how optical flow is used in image processing and computer vision. The importance of optical flow was highlighted by reviewing the wide range of applications it can be used in.

2.4 Summary

In this chapter a literature review for optical flow was presented. The aim of the literature review was to give an overview for variational optical flow algorithms. The chapter demonstrated the advances in the general optical flow equation (see Equation-2.1), and intensive discussion was presented to examine each part of that equation and the development it has undergone. More specifically a section was dedicated to discuss the data and the smoothness terms. In the section for the data term, several data terms were discussed and it was shown how this term developed in order to increase robustness against different types of noise, and robustness against illumination changes. Also this data term evolved to give more accurate estimation for the displacement field. The smoothness term also had its share of research, and more robust regularity was proposed, particularly smoothness terms that are more robust and that permits motion discontinuity at motion edges. This chapter also touched on several other topics such as research in implementation and real-time performance. Probabilistic formulation was also briefly introduced. Additionally an investigation for the application of optical flow in image processing was also introduced. The vast amount of application areas for optical flow show that optical flow algorithms are very important in image processing and computer vision fields.

In the literature two main issues can be identified. The first is related to image registration and optical flow. Optical flow was used for the purpose of image registration, both global and local optical flow algorithms were used. However there is a lack in the literature for the study of the effect of using combined local-global optical flow algorithms. The second issue is purely optical flow algorithm related. The total variation L^1 norm is known for its robustness and edge preserving performance. This norm was used in several algorithm for finding displacement fields. This norm is isotropic, this decreases efficiency of the estimated flow field near motion edges due to the reduced penalisation along these edges. In this thesis, those two issues are investigated and addressed.

Chapter 3

Local-Global Optical Flow For Image Registration

In the field of image processing, image registration is a closely related concept to optical flow. While the aim of optical flow is to calculate the displacement fields between two (or more) images taken with a time difference, image registration tries to align images so that the contents of these images can be matched. To accomplish this the displacement of each pixel between images is looked for, and correspondences between pixels in the two images must be established. These images can either be taken using the same camera with a time difference or taken using a different camera (view) at the same time. The alignment process can be described using a geometrical transformation (spatial mapping):

$$\mathbf{T} : I_B \mapsto I_A \Leftrightarrow \mathbf{T}(I_B) = I_A \tag{3.1}$$

where $\mathbf{T}$ is the spatial mapping, I_A is the source image (image to be transformed), and I_B is the target image. The concept of image registration can be illustrated as in Figure-3.1 next:

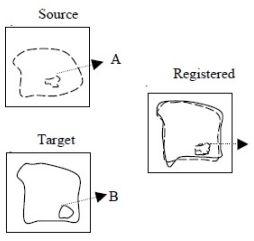


Fig. 3.1 Image registration illustration.
Correspondence between points A and B is found to perform registration

It is possible to classify image registration based on the way the spatial mapping function is estimated into two types. In the first type the spatial mapping can be found using a certain number of parameters to describe the mapping function, this is called parametric image registration. An example for this type of image registration is estimating the transformation based on descriptors detection and matching [8]. Alternatively the mapping function is found by estimating the displacement vector for each pixel, here the image registration is referred to as non-parametric image registration [7]. In the case of the latter type of image registration, utilising optical flow to estimate pixel correspondences becomes plausible.

Motion or deformation in optical flow is represented with velocity vectors originating from each pixel, specifying how each pixel moves between adjacent images. Periaswamy and Farid [135] used optical flow as a framework for image registration. Moreover [136] used optical flow to evaluate the result of medical image registration. The dense flow field calculated using the global optical flow methods allowed images with deformed contents to be registered. Hence, this can be found useful for the purpose of medical image registration [7]. A detailed description on the relation between image registration and optical flow can be found in the work of Lefébure and Cohen [129].

As a result of the relation between non-parametric image registration and optical flow, image registration performance inherits the shortcomings of the particular computational algorithm. In this chapter a new method for image registration is going to be used, this method is based on using a dense displacement flow field. Most of the image registration methods use either global or local optical flow methods. Bruhn et al [12] combined the local and global method to enhance the performance of the global method, and proposed a combined Local-Global optical flow (CLG). The CLG optical flow is used here for mono-modal registration, where both images are taken for the same scene with a time difference. It will be shown that this method is more robust compared to the global method of Horn-Schunck [1].

This chapter introduces the background theory and fundamental concepts in image processing used to estimate optical flow displacements. In the next section the Course-to-Fine framework is introduced with a detailed discussion on how it works with optical flow algorithms. Section-3.2 introduces further image processing techniques that are used to estimate optical flow, namely median and bi-lateral filters. Section-3.3 introduces the notion of structure tensor. Section-3.4 includes a discussion on the main benchmark datasets and means of quantitative assessment of optical flow algorithms. In Section-3.5 local and global variational optical flow computation is discussed as introduced by Horn-Schunck [1] and Lucas-Kanade [10], which is followed by a section introducing the Combined Local-Global method which is used in this thesis for image registration as introduced in [12]. Section-3.7

contains preliminary results of image registration using CLG optical flow. This is followed by a section that includes a proposed improvement on the CLG optical flow using bilateral filtering, replacing the Gaussian filtering used in the original CLG method. This final section is a summary to conclude this chapter.

3.1 Multi-scale Image Processing

One of the shortcomings of variational optical flow methods is that they can easily get trapped in local minima. This leads to undesired results especially in the cases where objects tend to have large displacements. In real life scenarios large displacements of objects' points are very much possible. To overcome this issue most optical flow algorithms perform minimisation in multi-scale (multi-resolution) images, where a pyramid of Coarse-to-Fine (C2F) images is created and used as a framework for the minimisation. In this section this framework is discussed in detail.

In the C2F framework, image sequences are coarsened several times via image re-sampling producing a hierarchy or a pyramid of images. This requires image re-sampling (up-scale or down-scale) multiple times. The solution is first looked for in the coarsest layer of this pyramid (top of the pyramid), where the problem is usually easier to be solved. This solution is then used to initialise a finer solution in the next layer of the image pyramid. This process is repeated for each pyramid layer until the original problem is solved. Multi-scale image processing is used in several applications, such as multi-scale image segmentation [137], object recognition [138], and detection [139]. In this section, an introduction for this framework is given, in addition to a discussion on the basic multi-scaling concepts used to create the C2F pyramid.

3.1.1 Spatial Filtering

Spatial filtering is a fundamental operation in image processing, it involves replacing the value of pixels with a weighted sum of its neighbours. In order to achieve this a *'correlation'* or a *'convolution'* is performed between the image and a certain filter *'kernel'*. The result is a modified version of that image. Depending on the kernel used, spatial filtering can be used to perform many image processing operations, such as image smoothing, edge detection, and finding image gradients, and also interpolation which is fundamental to image re-sampling

Correlation and convolution can both be used to perform spatial filtering, despite the fact that

they have some fundamental differences. Correlating an image with a kernel is finding the sum of the products between image pixels and kernel elements. This operation is repeated for each pixel while the kernel slides across the image dimensions. In a 2D discrete case (e.g. an image), correlation can be formulated as follows [140]:

$$I^{'}(x,y) = \sum_{i,j} h(i,\ j) I(x+i,\ y+j) \tag{3.2}$$

where $I^{'}$ is the result of the correlation and it is a modified version of the image I. Image indices are given by $x,\ y$, and $h(i,j)$ is a 2D correlation kernel. Figure-3.2 next demonstrates the sliding operation for the spatial filter:

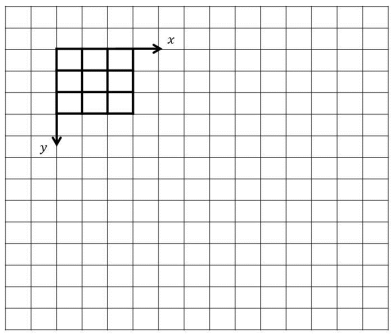


Fig. 3.2 Spatial filtering operation.
The filter kernel slides in the x and y direction to produce a filtered image, each pixel evaluated as the correlation of the image and the filter kernel overlaid over it.

Convolution is another mathematical operation used for spatial filtering, it has a similar application and working mechanism as the correlation. However an intrinsic difference is that the kernel is rotated 180^o before the convolution process [140]. In a 2D discrete case, convolution can be formulated mathematically as follows:

$$I^{'}(x,y) = \sum_{i,j} h(x-i,\ y-j) I(i,\ j) \tag{3.3}$$

which can be also written in the following form [140]:

$$I^{'}(x,y) = \sum_{i,j} h(i,\ j) I(x-i,\ y-j) \tag{3.4}$$

where $h(i, j)$ is known as the convolution kernel or filter mask, and (i, j) are the indices of that kernel. Hence, image pixels are determined by weighted values of its neighbourhood. To demonstrate a simple example of the difference between correlation and convolution operations, consider Figure-3.3 next which depicts a 2D matrix which can be considered as a segment of an image. The figure also depicts a random kernel h.

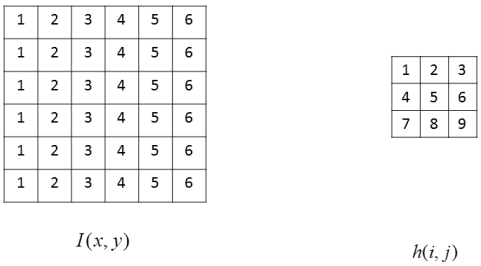


Fig. 3.3 An image and a kernel.
Left: A segment of an image with each square representing a pixel. Right: An example of a filter kernel, this kernel is applied to each pixel of the image.

Figure-3.4 next demonstrates the results of a correlation and a convolution processes:

43	82	121	160	199	133
51	96	141	186	231	150
51	96	141	186	231	150
51	96	141	186	231	150
51	96	141	186	231	150
25	46	67	88	109	67

17	38	59	80	101	87
39	84	129	174	219	180
39	84	129	174	219	180
39	84	129	174	219	180
39	84	129	174	219	180
35	74	113	152	191	153

Fig. 3.4 The result of a correlation and a convolution operations
Left: Result of correlation. Right: Result of convolution.
Although the image and the kernel are the same, the outcomes of correlation and convolution are different, this is because the kernel is rotated by 180^o before applying convolution.

Figure-3.3 depicts a segment of an image $I(x, y)$, where each small square represents a pixel. Numbers inside the pixels are the intensity levels for each pixel. Figure-3.3 also include a kernel $h(i, j)$. Figure-3.4 depicts the results of applying the kernel to the segment of the image using convolution and correlation. Despite the fact that the segment of the image and the kernel are the same, the results apparently differ; this is due the rotation of the kernel by 180^o before performing the convolution.

As was pointed out earlier, spatial filtering can be used for many image processing applications. Many different filter kernels can be used depending on the application required. Spatial filtering can be used for image smoothing, for example a Gaussian kernel can be used as a low pass filter. In the continuous case a 2D Gaussian function $K(x,y)$ can be given by the following equation:

$$K(x,y) = \frac{1}{\sigma^2 2\pi} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.5}$$

where σ is the standard deviation. An example of the discrete version of the Gaussian function being used as a low pass filter is shown in Figure-3.5 for a 5×5 approximation of a Gaussian kernel.

$$\frac{1}{250}\begin{bmatrix} 8 & 9 & 10 & 9 & 8 \\ 9 & 11 & 12 & 11 & 9 \\ 10 & 12 & 12 & 12 & 10 \\ 9 & 11 & 12 & 11 & 9 \\ 8 & 9 & 10 & 9 & 8 \end{bmatrix}$$

Fig. 3.5 A 5×5 Gaussian kernel approximation with with $\sigma = 3$.
This Gaussian filter can be used as a low pass filter.

Applying spatial filtering using a Gaussian kernel smooths the image by suppressing high frequencies such as noise and edges, therefore the resulting image appears to be blurry. Two main parameters affect the performance of this kernel these are the standard deviation σ and the size of the kernel itself. With fixed kernel size the higher the standard deviation for the kernel the more blurry the resulting images will look, this is due to close weights assigned to pixels in the neighbourhood. Figure-3.6 demonstrates the effect of different σ on the resulting image. In this context it is worth pointing out that convolution and correlation gives the same results, this is due to the symmetry that such filter kernels have (see Figure-3.5).

Spatial filtering can also be used to find image gradients. For a 2D image, gradients can be defined as follows:

$$\nabla I = [\, I_x,\, I_y \,] = [\, \frac{\partial I}{\partial x},\, \frac{\partial I}{\partial y} \,] \tag{3.6}$$

The basic definition for image derivatives are:

Fig. 3.6 Smoothing an image using a Gaussian Kernel.
It can be seen that the image becomes more blurry with the increase of σ
Top left: Original image. Top right: $\sigma = 0.5$. Bottom left: $\sigma = 1$. Bottom right: $\sigma = 3$.

$$\frac{\partial I}{\partial x} = I(x+1,y) - I(x,y) \tag{3.7}$$

$$\frac{\partial I}{\partial y} = I(x,y+1) - I(x,y) \tag{3.8}$$

According to this equation, the value of the image derivative at a certain pixel is equal to the difference in intensity level of that pixel and the one next to it. These equations can be formulated as a spatial filtering operation by using the following kernel for example to obtain the horizontal derivative:

1	-1

Fig. 3.7 Filter kernel to find the horizontal derivative of an image

In the same manner it is possible to obtain the derivative in the y direction, using the transpose of the kernel in Figure-3.7 above. In the following figure, an example of using that kernel to obtain image derivatives is shown.

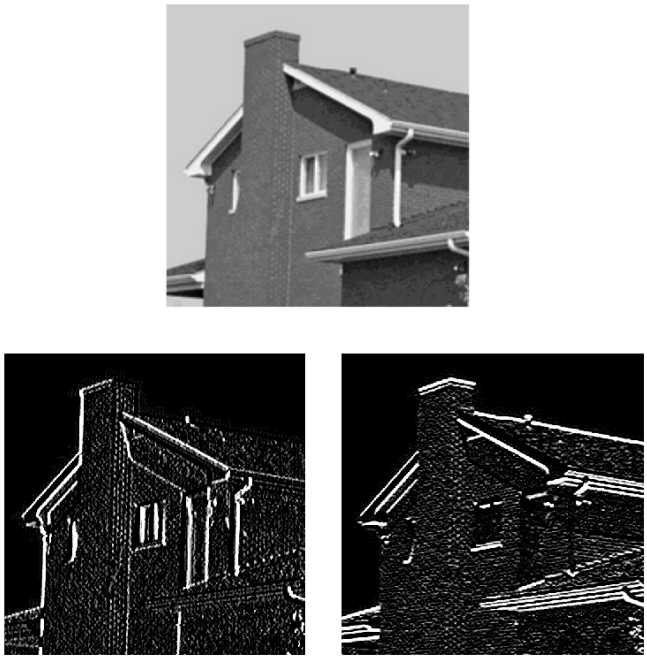


Fig. 3.8 Image derivatives in the horizontal and vertical direction.
Top: Original image. Bottom left: Derivative in the x-direction. Bottom right: Derivative in the y-direction

Several other filter kernels can be used to find image gradients, such as the central point difference $[-1,\ 0,\ 1]$. More robust filter kernels have also been used, such as the filter kernel in [12]:

$$\frac{1}{60}\begin{bmatrix} -1 & 9 & -45 & 0 & 45 & -9 & 1 \end{bmatrix}$$

Fig. 3.9 A robust convolution derivative kernel was used in [12] to find image gradients.

The notion of image smoothing and filtering finds direct application in variational image processing algorithms, for example in CLG optical flow [12] image smoothing was used to increase the robustness of the optical flow energy equation. Smoothing an image via spatial

filtering can be thought of as replacing the intensity value of a pixel with a weighted sum of the pixel's neighbourhood, hence the effect of noise can be mitigated.

3.1.2 Up-scaling Images

Up-scaling can be thought of as overlaying an image over a grid with higher resolution, this is done via up-sampling the original image. This operation produces pixels that have no information. Values for pixels with no information are estimated via interpolation from neighbouring pixels in the original image. For example consider a rectangular image with dimensions $N \times M$. This image is required to be up-scaled by a certain factor producing an image with dimensions $rN \times rM$, where r is the up-scaling factor, a certain number of pixels will be inserted between each two adjacent pixels in the original image (Figure-3.10).

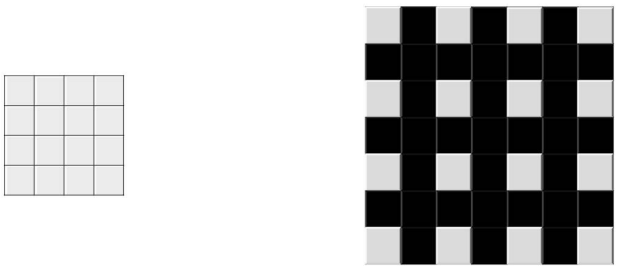


Fig. 3.10 Image Up-scaling.
Left: Original image with dimensions $N \times M$. Right: Up-sampled version image where additional pixels were inserted.

The interpolation method can be formulated as a spatial filtering operation, for example the interpolation can be given as a convolution process (Equation-3.3, 3.4), [141]:

$$Iu = \sum_{x,\, y} I(x,\, y)h(i - rx,\, j - ry) \tag{3.9}$$

where Iu is the up-scaled image. Several interpolation kernels can be utilized to interpolate the zero pixels, the choice of the kernel is an important part of the up-scaling process. For example pixel values can be interpolated directly from neighbouring pixels, this is known as nearest-neighbour interpolation. This interpolation method is perhaps the most basic

one, often used when speed of implementation is required. However this method is not very efficient and the output image suffers from a block-like appearance, which becomes obvious at edges.

Another type of interpolation is the bilinear interpolation, here the interpolated pixel value is derived from the values of the 4 surrounding pixels, and is found as a weighted average for these pixels. In the case of bilinear interpolation, the interpolated pixels have a better approximation value in comparison to nearest-neighbour interpolation, and the transition between intensity (or colour) values is smoother. In mathematics, bilinear interpolation is an extension of linear interpolation in a 1D setting, where a value of an interpolated point is found to be lying on the straight line connecting two points. In the 2D case this is extended to 4 points (pixels), and the intensity value of the interpolated pixel is found as a linear combination of the 4 surrounding pixels [142]. Let the four surrounding pixel have the following coordinates, $I^0(x_0,y_0)$, $I^1(x_1,y_1)$, $I^2(x_2,y_2)$, $I^3(x_3,y_3)$. Then the intensity value of the up-scaled image at a certain pixel is [142]:

$$I(x,y) = A + Bx + Cy + Dxy \tag{3.10}$$

where A, B, C, D are constants. which can be found by solving the following equations:

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 \\ 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \end{bmatrix}^{-1} \begin{bmatrix} I(x_0,y_0) \\ I(x_1,y_1) \\ I(x_2,y_2) \\ I(x_3,y_3) \end{bmatrix} \tag{3.11}$$

Bi-cubic interpolation includes even more pixels in the calculation than the bilinear interpolation, a 4×4 neighbourhood is considered in the calculations. Bi-cubic interpolation is an extension for the cubic interpolation in 1D. It can be easily found by applying the following cubic convolution kernel [141]:

$$h(k) = \begin{cases} 1-(a+3)x^2+(a+2)|x|^3 & if|x|<1 \\ a(|x|-1)(|x|-2)^2 & if1 \leqslant |x|<2 \\ 0 & elsewhere \end{cases} \tag{3.12}$$

where a specifies the derivative at $x = 1$, usually chosen to be -1 or -0.5.

The aforementioned up-scaling algorithms are the most common methods used in creating

an image pyramid for optical flow. The performance of these methods differ especially when comparing the speed, sharpness of the interpolated image and how well they can preserve details and edges. Figure-3.11 depicts an example image that is going to be up-scaled five times using the three previously discussed interpolation methods. Figure-3.12 next depicts the differences between the three interpolation methods.

Fig. 3.11 Image *'lena'* to be up-scaled 5 times.
The area inside the red square is shown in the next image.

Figure-3.12 next shows the part of the image that is enclosed inside the red square enlarged 5 times.

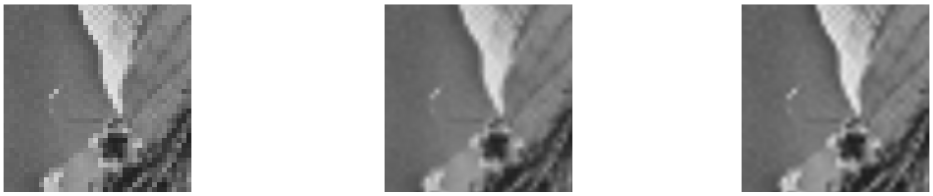


Fig. 3.12 Comparison of the three interpolation kernels.
Left: Nearest-neighbour. Middle: Bilinear. Right: Bi-cubic.

The differences between the three interpolation methods can be seen in Figure-3.11, and Figure-3.12. The image enlarged using the nearest-neighbour interpolation method is pixelated and the transitions between different intensity values are obvious in areas with salient structures, this interpolation method is usually used when a fast computation is required as pointed out earlier. Images enlarged using the bilinear and bi-cubic methods look more similar compared to the nearest-neighbour one, and they both show finer and smoother results,

however the bilinear one has lost a bit of sharpness and looks blurry compared to the other two images.

3.1.3 Down-scaling (Decimation)

Down-scaling gives an opposite result to that of up-scaling, where an image is down-sampled to obtain another version of the image with less resolution. The resulting image will have smaller dimensions, obviously this means that there will be loss of information. In practice to perform down-scaling, every *n*th pixel in the image is replaced with a weighted sum of the surrounding pixels, while the rest of the pixels are eliminated [141]. Similar to up-scaling several kernels can be used in the decimation process, and the down-scaling can be formulated as a convolution process. However the kernel in this case is a stretched and re-scaled version of the interpolation kernel previously discussed. This can be noticed in the following equation [140], [141]:

$$Id = \sum_{i,j} I(i,\ j)h(ni-k,\ nj-k) \tag{3.13}$$

where Id is the down-scaled image and n is the down-scaling ratio. Figure-3.13 next shows the image in Figure-3.11 down-scaled by a factor of 2 using the three kernels discussed in the previous section.

Fig. 3.13 Comparison of the three Down-scale interpolation kernels.
Left: Nearest-neighbour. Middle: Bilinear. Right: Bi-cubic.

To show the the down-scaling effect more clearly, Figure-3.14 demonstrates the image 'lena' down-scaled and then up-scaled with a factor of 2.

Fig. 3.14 Comparison of image resizing with factor of 2.
Left: Image down-scaled then up-scaled via nearest neighbour. Middle: Image down-scaled then up-scaled via bilinear interpolation. Right: Image down-scaled then up-scaled via bi-cubic interpolation.

Performing down-scaling usually generates aliasing artefacts in the down-scaled image. The aliasing artefacts are due to the existence of high frequency details in the image which is to be projected on a smaller resolution grid; therefore a low-pass filter is used prior to the down-scaling process to suppress such high frequencies in the original image.

Resizing images is a fundamental process in multi-scale image processing. As the name implies, multi-scale or multi-resolution image processing requires the processing of the image at different scales. To this end, image down-scaling is used to find a coarser (smaller) version of the image to be processed. Later the solution found needs to be up-scaled to obtain a finer version of it.

3.1.4 Image Warping

Given an image and a transformation map it is possible to deform the image according to the given mapping. This operation belongs to the geometrical transformation process in image processing. In general image warping techniques can be classified into two types, the first uses a number of parameters to create a global transformation model to perform warping, while in the second type the transformation model is available [8]. If the transformation (displacement) or deformation vector is available for each pixel in the image, it is easy to produce a warped image. Consider an image I with a transformation mapping in the form of a displacement vector for each pixel u and v. In the warping process, each pixel in the warped image gets the value of the corresponding pixel in the original image. This can be

expressed as follows:

$$I(x,y) = I_{warped}(x+u,\ y+v) \tag{3.14}$$

Figure-3.15 depicts this operation.

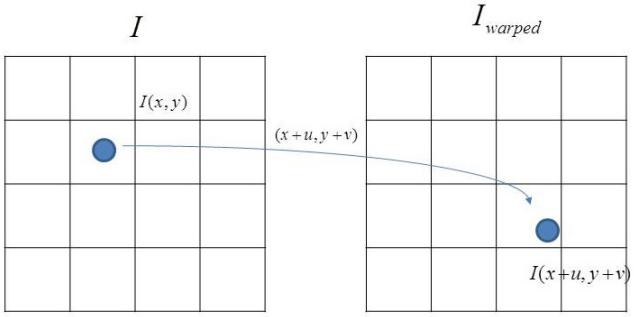


Fig. 3.15 Forward Image Warping.

This warping technique is called the *'forward warping technique'*. The values of u and v are not always integers, therefore these values must be rounded to obtain pixels residing on the grid in the warped image. This process may produce holes and overlaps in the resulting image. To mitigate this an alternative method for warping is used which is called the *'backward warping technique'*. In this technique pixels are backward mapped from the grid representing I_{warped} towards I. Additionally the values for pixels colours are determined via interpolation with a certain kernel. This reduces holes and overlaps creation. [8], [141].

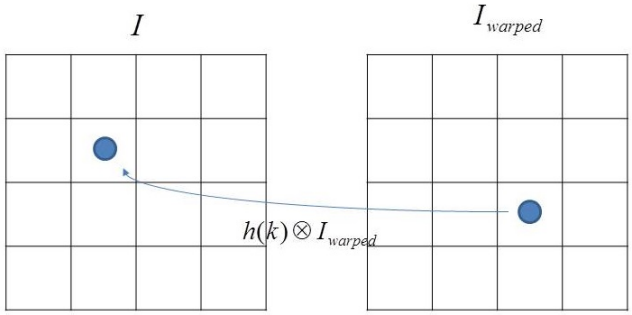


Fig. 3.16 Backward Image Warping

To demonstrate the difference between the two warping methods, consider the following image in Figure-3.17. In this figure a second image in a sequence is warped using the dense

optical flow field obtained using the global Horn-Schunck algorithm [11].

Fig. 3.17 Forward and Backward warping.
Left column: Forward image warping, Right column: Backward image warping.
Image sequence:MiniCooper and Grove2 [6]
Cracks in the image are visible in the forward-mapped images

The following figure is a magnified segments of forward-warped images.

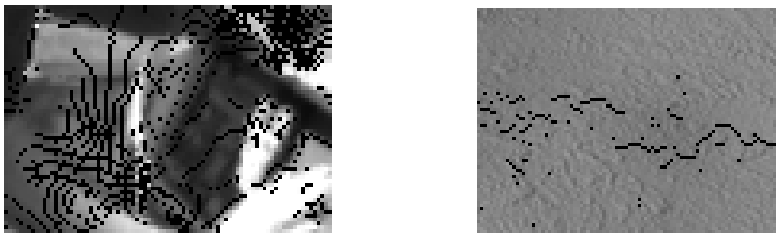


Fig. 3.18 Magnified forward-warped images segments.
Cracks are apparent in images with forward mapping.

The process of image warping is fundamental to many image processing application. For example it is an essential part of image registration, where the aim is to align two images. This alignment obviously involves deforming or warping one image to align with the other. It also plays a vital role in the C2F framework, which in turn is essential to variational image processing techniques, including variational optical flow algorithms. In such algorithms, multi-scale image pyramids are produced and optical flow displacements are estimated for

each layer. At each layer of the pyramid, one image is warped towards the other using the estimated displacement fields in the previous layer.

3.1.5 Coarse-to-Fine Framework (C2F)

The aforementioned techniques and concepts can now be used to create an image pyramid, where an image is coarsened several times as can be seen in Figure-3.19 next.

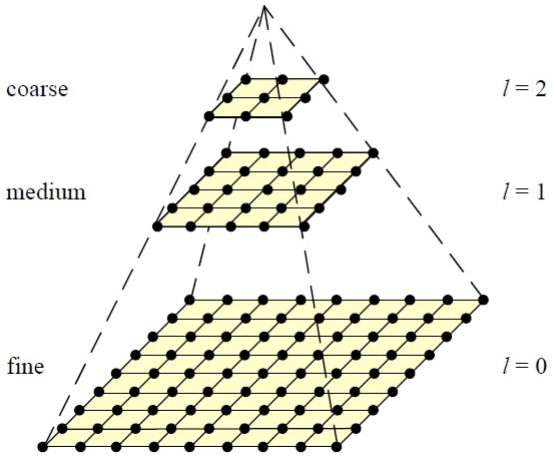


Fig. 3.19 Multi-scale image pyramid.
The coarse layer is obtained by down-sampling the previous layer. Image taken from [141]

The production of the pyramid in the figure above can be illustrated as a pseudo-code as can be seen in Algorithm-1. Usually each pyramid layer dimensions are reduced by a factor of 2, this should be sufficient to compute optical flow using the Horn-Schunck algorithm. However, some algorithms require a finer pyramid, Brox et al. [13] used a very fine pyramid with a down-scaling factor given by $0.95^{(L-l)}$, where l is the current layer and L is the number of layers, chosen to be smallest resolution where image gradients can be found.

Algorithm 1: Creating an image pyramid

Input: Image I, number of pyramid layers L, current layer l
initialization;
while $l \leqslant L$ **do**
    Down-scale I by a factor of $2l$;
    produce I_l;
end

To calculate optical flow under C2F using any variational algorithm, the displacement field

is calculated in the coarsest layer of the pyramid. This is done between the coarsest versions of the two images I_1^l and I_2^l, where l is equal to 0 at the base of the pyramid. The displacement field components calculated at a certain layer is denoted here by (du, dv), and is initialised to $(\mathbf{0}, \mathbf{0})$ at the top of the pyramid. These displacement fields calculated at a certain layer are used as an initialisation for the calculation at the next finer layer. Therefore the displacements fields are up-scaled via interpolation using any of the aforementioned methods. In addition to that I_2^l at $l = 2$ is warped towards I_1^l. At $l = 2$ the displacement fields are calculated in the same manner, and the final displacement field at that layer is found to be equal to the sum of the current displacement (du, dv) and the displacement calculated in the previous layer (u_{l-1}, v_{l-1}). This process continues until the original image at the base of the pyramid is reached. Algorithm-2 depicts a general framework for optical flow estimation under C2F.

A typical pyramid can be composed of 4 layers, where the dimensions of an image in each layer is half of that in the lower layer. This pyramid is sufficient to produce good results [143] for the Horn-Schunck algorithm. Figure-3.20 next includes a demonstration for a 4 layer image pyramid of the RubberWhale sequence [6].

Figure-3.21 demonstrates the optical flow field for several image sequence. For each sequence optical flow was calculated twice. One result is obtained using C2F and one without C2F. The results obtained without the use of C2F suffers from being trapped in local minima, while the results obtained via the use of the C2F framework appear to be more accurate.

Algorithm 2: Calculating optical flow under C2F framework

Input: Images I_1, I_2;
Number of pyramid layers L, current layer l
initialization;
initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;
while $l \leqslant L$ **do**
    Up-scale size of (u_{l-1}, v_{l-1}) to size (u_l, v_l);
    Warp I_{2l} towards I_{1l};
    Find (du, dv);
    Calculate $(u_l, v_l) = (u_{l-1}, v_{l-1}) + (du, dv)$;
end
Output: (Displacement field (u, v))

Fig. 3.20 A 4 layer image pyramid.

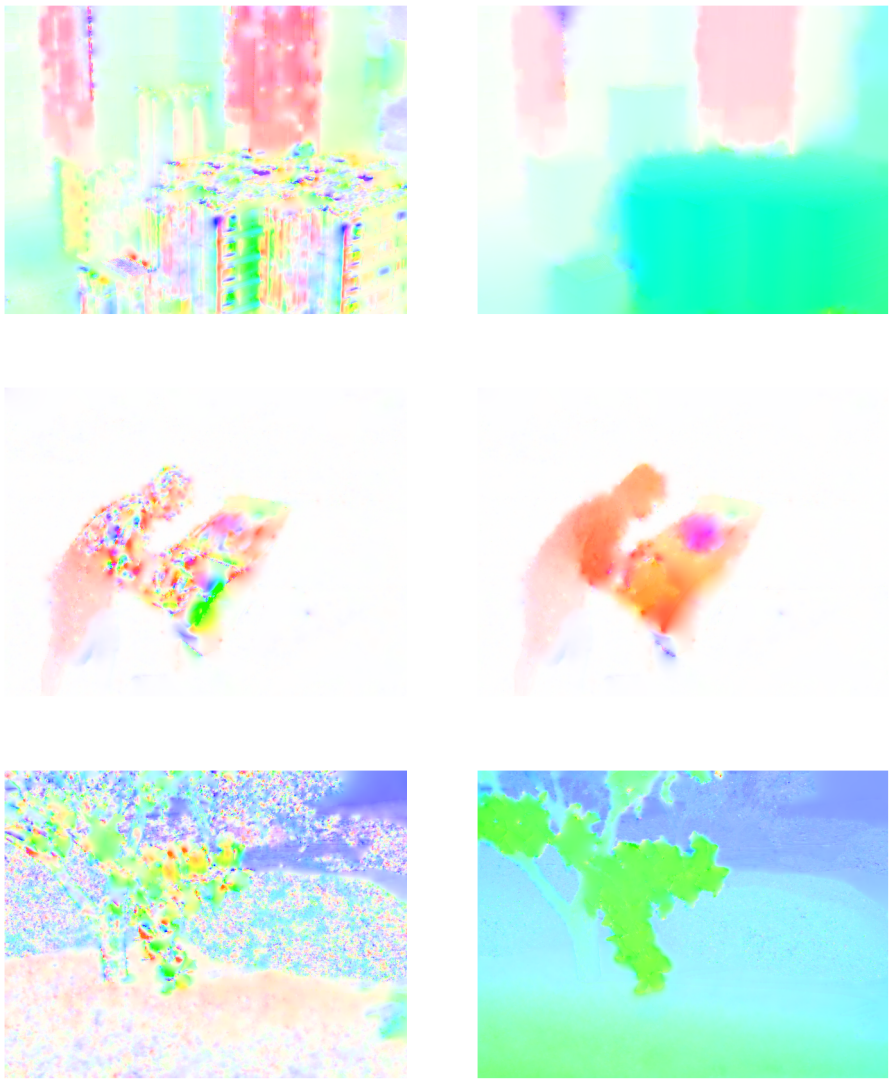


Fig. 3.21 A comparison of Horn-Schunck optical flow with/without C2F framework. Left Column: Optical flow obtained without C2F. Right column: Optical flow obtained with C2F.
The displacement field estimated without the use of C2F suffered being trapped in local minima.

3.2 Other Types of Filtering

In the previous sections, the notion of image filtering was discussed. Image filtering was discussed as an application of correlation and convolution. It was also shown that smoothing

images using certain kernels (e.g. Gaussian) had a blurring effect on the smoothed images. Since edges are important information contained in images, it was necessary to develop smoothing techniques that would preserve edges during the smoothing process. This also applies to optical flow algorithms, where it is necessary to preserve image motion boundaries in the calculated displacement field.

The blurring effect that appears in the motion boundaries of the estimated optical flow field can be attributed to several factors, the first is the assumption of global smoothness in the displacement field, this assumption is violated at motion boundaries. If the smoothness term penalises this assumption severely the results is a blurred motion boundaries. The blur can also be found in CLG algorithms for example, where the value of pixels are replaced with a weighted average of the pixel values surrounding that pixel, which produce this blurry effect when pixels are near image boundaries. A possible solution is to replace the linear filter with a non-linear filter that respects the presence of boundaries in an image. In this section additional types of filters are discussed, these filters have an immense number of applications in image processing, including applications of variational optical flow. In the following subsection the median and bi-lateral filters are discussed.

3.2.1 Median Filter

A median filter is a spatial non-linear filter, it belongs to what is known as *'Order-Statistics filters'* [140]. Unlike linear spatial filtering discussed earlier these types of filters do not use convolution, instead they rely on exploiting statistical properties in a certain neighbourhood. As the name implies, a median filter replaces pixels with the median of pixel intensities inside a certain neighbourhood. Other examples of statistical filters are the max filter which replaces the intensity value of a pixel with the maximum in a neighbourhood, the min filter does the same but using the minimum value.

Consider a set of N numbers $A = a_1, a_2, ...a_N$ sorted either in ascending or descending order, the median of such a set is the number lying in the middle of the set if N is odd, and is the average of the two numbers lying in the middle of the set if N is even. The median filter may outperform linear filters in some cases especially in terms of preserving edges. It is useful primarily when the noise appears suddenly and discontinuously in the image such as *Impulse* noise also referred to as *salt-and-pepper* noise [140], where the value of the pixel with such noise will be replaced by an intensity value from within its neighbourhood. Figure-3.22 illustrates the performance difference between a median filter and a linear Gaussian filter.

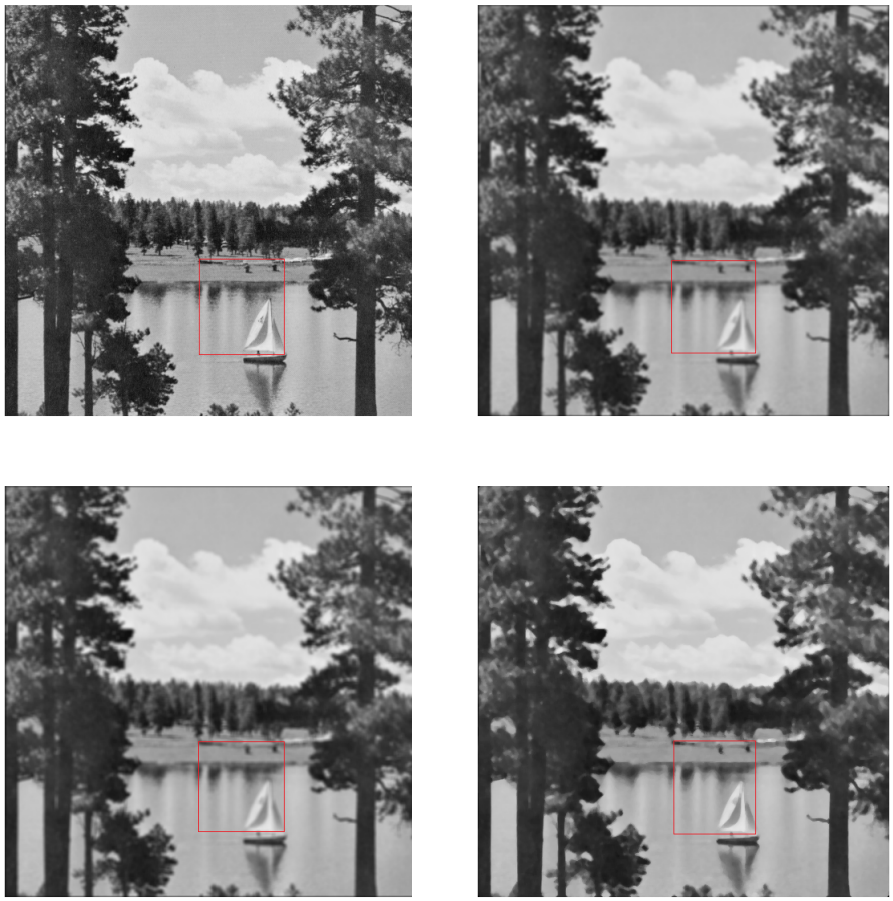


Fig. 3.22 Smoothing an image using median filter and two Gaussian filters.
Up left: original image. Up right: Gaussian image smoothing with $\sigma = 2$. Bottom left: Gaussian image smoothing with $\sigma = 5$. Bottom right: Image filtering using median filter.

To further examine the difference, the following figure is a magnified copy of the previous figure. The comparison is done between the median filtering effect and one of the Gaussian filters. It can be seen that the image filtered with the median filter, although it looses some sharpness, the details and edges are clearer and less blurry.

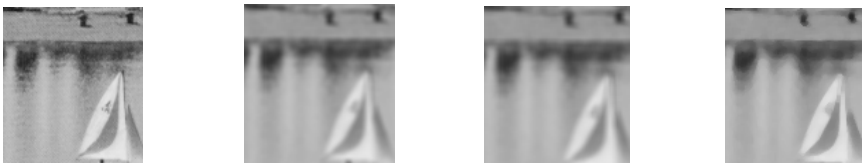


Fig. 3.23 Filtering comparison.
Left: segment of the original image. Middle left: segment of the image filtered with Gaussian of $\sigma = 2$. Middle right: segment of the image filtered with Gaussian of $\sigma = 5$. Right: segment of the image filtered with a median filter.

As a consequence of the performance of this filter in the presence of an impulse-like noise, the median filter was integrated into the C2F framework for optical flow calculations [14] in what is known as *'Splitting methods'* [24], where optical flow is calculated in a dual minimisation step (see Section-4.1). The use of a median filter as an intermediate filtering step improved the performance of the optical flow algorithms. After finding the displacements fields in each pyramid layer, a median filtering is applied to the calculated optical flow. This ensures the removal of outliers that may appear during the calculation of the flow field displacement. Figure-3.24 next demonstrates the optical flow fields for two image sequences of the Middlebury dataset [6]. The displacement field is estimated via the $TV-L^1$ algorithm, with the implementation of [14] downloaded from their website[1].

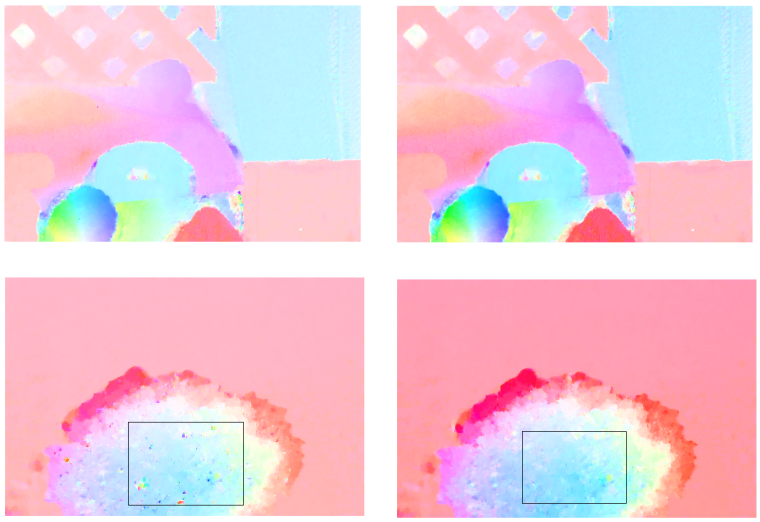


Fig. 3.24 The median filter effect on the calculation of optical flow field.
Left column: optical flow computed without median filter. Right column: optical flow computed with median filter.
Areas enclosed in the black squares in the lower row shows outliers removal after using median filter.

In Figure-3.24 the difference in the two flow fields is noticeable, where one of the flow fields was computed without the use of median filtering, while the other used that filter as an intermediate step. It can especially be noticed in the 'Hydrangea' sequence in the lower row of images. The displacement flow field calculated without the median filter clearly shows outliers, while the one with the median filter shows the absence of those outliers. In addition to removing outliers Sun et al. [143] showed that the inclusion of median filtering produce higher energy results.

[1] http://gpu4vision.icg.tugraz.at/index.php?content=downloads.php

3.2.2 Bi-lateral Filter

The bi-lateral filter is known for its edge-preserving properties [59], [60], however the working mechanism is completely different compared to the previously discussed median filter. It can be seen as a Gaussian filter that respects boundaries, and it can be thought of as a combination of more than one technique of filtering. As was pointed out earlier during the discussion of spatial filtering, Gaussian kernels for convolution introduces blurring across edges. This is attributed to the fact that each pixel is replaced with values that are a weighted average of pixels in the neighbourhood, and if this pixel resides near an edge its value is going to be affected by the pixel across the edge, which usually have very different values. The bi-lateral filter addresses this issue by adding weights to pixels depending on their intensity values and how close these values are to the intensity of the pixel being considered.

In the linear spatial filters discussed earlier (e.g. Gaussian), the weight for each pixel in the window is a function of the spatial distance from the centre pixel. The previously discussed linear filtering (Equation-3.4, Equation-3.3) can be expressed as follows:

$$I^{'}(\mathbf{x}) = \frac{1}{w_g} \sum_{\mathbf{i}} K(\mathbf{i},\mathbf{x}) I(\mathbf{i}) \tag{3.15}$$

where $\mathbf{i} = (i,\ j) \in \Omega_D$, Ω_D is the neighbourhood region specified by the kernel size, K is the Gaussian kernel, w_k is a normalising term, and can be defined as:

$$w_g = \sum K(\mathbf{i},\mathbf{x}).$$

Similarly it is possible to formulate a filter that gives weight based on the similarity in intensity values, such a filter can be defined as:

$$I^{'}(\mathbf{x}) = \frac{1}{w_s} \sum_{\mathbf{i}} S(\ I(\mathbf{x}),\ I(\mathbf{i}))\ I(\mathbf{i}) \tag{3.16}$$

where $s(I(\mathbf{x}),\ I(\mathbf{i}))$ determine the weight depending on the similarity between the two pixels, w_s is also a normalising term, and can be given as:

$$w_s = \sum S(\ I(\mathbf{x}),\ I(\mathbf{i})\ )$$

The bi-lateral filter combines those two filters to produce a filter with weights change in

spatial and also intensity (or colour) domains. Such a filter can be expressed as follows:

$$I^{'}(\mathbf{x}) = \frac{1}{w} \sum_{\mathbf{i}} S(\, I(\mathbf{x}),\, I(\mathbf{i}))\, K(\mathbf{i},\mathbf{x})\, I(\mathbf{x}) \tag{3.17}$$

where w here is the normalising term, and can be defined as:

$$w = \sum S(\, I(\mathbf{x}),\, I(\mathbf{i}))\, K(\mathbf{i},\mathbf{x}) \tag{3.18}$$

In practice S and K both can have Gaussian distribution with different standard deviation value. Hence, this filter gives different weights to pixels in the window. The weight varies as a function of two variables, the first is the spatial distance of that pixel to the centre pixel, the bigger the distance the smaller the weight given. The second variable is the intensity level similarity, the closer the intensity pixel values to the considered pixel, the higher the weight assigned. This reduces blurring, as pixels across edges are expected to have different intensity or colour values. Both kernels can have a Gaussian distribution. Figure-3.25 shows an illustration of the behaviour of a bi-lateral filter near an image edge. As can be seen in the figure, the spatial Gaussian filter weight is decreased as pixels are far from the central pixel, while the bi-lateral filter kernel weight decreases rapidly at the existence of image edge.

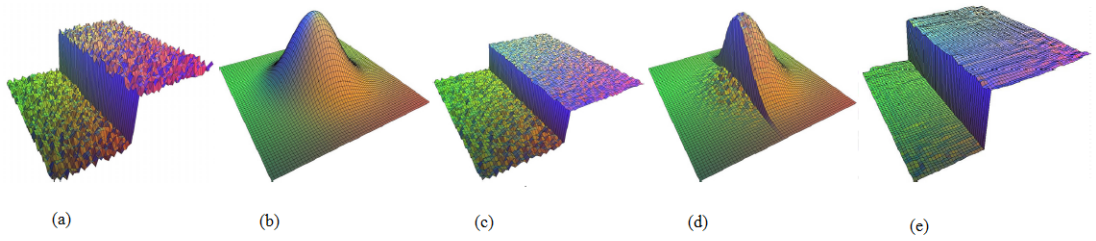


Fig. 3.25 Bi-lateral filtering.
(a) Input image. (b) Spatial Gaussian kernel $g(\mathbf{i},\mathbf{x})$. (c) Intensity similarity kernel. (d) Combined kernel (e) Output. [Image taken from [60]]

Figure-3.26 next demonstrates some examples for smoothing images using a bi-lateral filter, while Figure-3.27 shows a segment of an image providing more illustration.

Fig. 3.26 Smoothing using Bi-lateral filter.
Left column: original images. Right column: images smoothed via bi-lateral filtering. Despite the smoothing effect of the filter, the image edges are still visible, this can be noticed for example in the clouds in the 'lake' image.

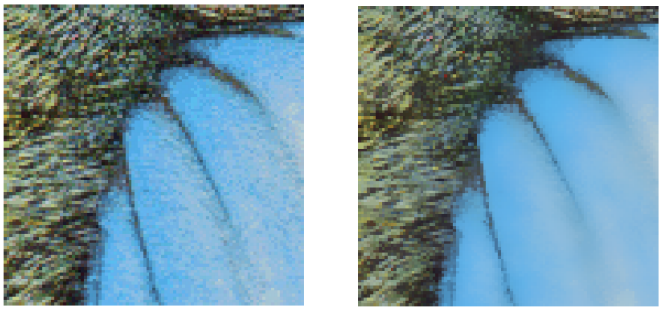


Fig. 3.27 An image segment demonstrating smoothing using Bi-lateral filter.

The bi-lateral filter has been used by some optical flow algorithms to improve the performance at the presence of edges and object boundaries [4] [106].

3.3 Structure Tensor

A structure tensor matrix is a fundamental tool used in image processing and computer vision, it is also known as the second moment matrix [144]. It gives an idea of the dominant orientation of gradient in a certain neighbourhood, and it is derived from image gradients, therefore it has been widely used in several applications such as corner detection [145], [146], texture analysis [147], [148], [149], and optical flow calculation [12]. An initial structure tensor matrix J_0 of a 2D image I can be written as follows:

$$J_0 = \nabla I \nabla I^T = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{3.19}$$

In the formulation of structure tensors, it is desired to include information in a certain neighbourhood around each pixel. To this end the structure tensor J of a 2D image can be found by convolving J_0 with a Gaussian kernel K_ρ:

$$J = K_\rho * J_0. \tag{3.20}$$

This formulation can be easily extended to higher dimensions. Although the information provided by the structure tensor matrix is derived from the image gradients, it offers more useful information due to the smoothing process via the Gaussian kernel with a certain integration scale ρ. This allows to examine the orientation and magnitude of a structure in a certain neighbourhood. In addition to that the squaring operation can help to avoid cancellation of gradients with opposite direction during the smoothing process [144].

The importance of the structure tensor matrix lies in the orientation information it conveys, which can be obtained via the calculation of the eigenvalues and eigenvectors. It is known that for a certain matrix (let it be J in this case) eigenvalues and eigenvectors can be calculated, and that:

$$J\mathbf{e} = \lambda \mathbf{e} \tag{3.21}$$

where $\mathbf{e}$ are the set of eigenvectors ($\mathbf{e}_1$, $\mathbf{e}_2$...$\mathbf{e}_n$), and λ are the corresponding eigenvalues (λ_1, λ_2,... λ_n), and n is the number of dimensions of the square matrix J. Equation-3.21 can

also be written as follows:

$$J = \mathbf{e}^{\top} \lambda \mathbf{e} \tag{3.22}$$

In image processing, the structure tensor computed for a 2D image is useful to give an idea of the dominant orientation in the neighbourhood. The two eigenvectors extracted from the 2×2 matrix are one pointing across the dominant orientation, while the other points along the dominant orientation. For example if the pixel is near an edge where the gradient is high, the first eigenvector points vertical to the edge while the other is parallel to that edge [40]. Coherence is another useful piece of information that can be extracted from the structure tensor and its eigenvalues/eigenvectors, usually defined as the largest number of eigenvalues divided by the smallest [144].

The information extracted from the structure tensor matrix can be used to determine the dominant orientation in a local neighbourhood. The eigenvalues are proportional to the magnitude of the gradient orientation and thus can be found useful in edges and corners detection [145].

3.4 Benchmark Datasets and Error Measures

In this section, a general discussion on the error measurements to assess the estimation accuracy of optical flow fields is introduced. As optical flow algorithms were continuously improving, it was necessary to develop error measures to compare the performance of different algorithms, this comparison could either be qualitative or quantitative. Most common approaches for quantitative performance measurements calculates error in relation to a precomputed ground truth displacement field. A qualitative measure on the other hand relies on visually inspecting a colour-coded version of the computed flow field. While it may be more efficient to quantitatively compare performance of different algorithms, it is not always possible to compute the ground truth of test images. Depending on their source, test images can be divided into two types, synthetic and real world images. It is relatively easy to find a precise ground truth for synthetic images, as these images are generated by computer graphics. Conversely, obtaining ground truths for images with real objects are more challenging.

3.4.1 Obtaining Image Sequences and Ground Truth

Middlebury dataset

Different techniques were followed to obtain such ground truths, an example of this is the Middlebury dataset [6], which includes synthetic and non-synthetic images. Since this dataset will be the most used dataset in this thesis, a more detailed discussion for this dataset is included here. The dataset is divided into four types of test images, each encompassing different challenge goals, such as non-rigid motion, real world scenarios, and dense ground truths with sub-pixel accuracy. The four types of images are as follows:

- Non-synthetic images with non-rigid moving scene: To obtain these images a special set was built on a computer-controlled stage. This set is moved in small steps, and two types of images are taken at each step, one using ambient light while the other is taken under UV lighting. The combination of UV with special paint applied to the set helps to preserve textures of the objects in the scene. The displacement is then found by a local search in a small window in the images taken under UV lights. Sub-pixel accuracy is obtained using the Lucas-Kanade [10] algorithm. One advantage of such kind of test images is the use of real cameras to obtain these images, which means natural illumination conditions. On the other hand the images taken in a laboratory are not real world scenarios. Figure-3.28 includes examples of such images.

Fig. 3.28 Real (Non-Synthetic) image examples from the Middlebury dataset. [6]. Left: Frame-10 of ‘Army’ sequence. Right: Frame-10 of ‘Mequon’ sequence.

- Synthetic image: These images are generated using computer graphics. The advantage of such images is the ability to compute a precise ground truth, in addition to having control of the texture and the scenario of the scenes. Two types of images were generated using special software. The first type is natural scenes with non-rigid

motion (e.g. trees). The second is generated 'Urban' scenes with building of different shapes and textures.

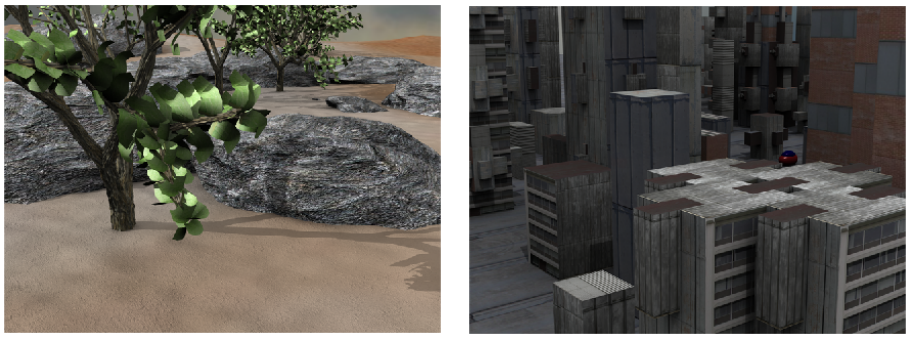


Fig. 3.29 Synthetic image examples from the Middlebury dataset. [6].
Left: Frame-10 of 'Grove2' sequence a natural scene. Right: Frame-10 of 'Urban2' sequence an urban structures scene.

- Images via interpolation: These images do not have ground truth. They are used to check the optical flow algorithms by how well they can predict intermediate frames. Images for these datasets are taken using a camera with a 60 frames/second rate. Every other frame was chosen to be used for testing, while the intermediate frames were kept to be used as ground truth. Indoor and outdoor images were taken, including different objects such as people, moving vehicles and some urban structures. To test the optical flow algorithms, intermediated frames are generated via interpolation and compared with the omitted intermediate frame. Figure-3.30 demonstrate examples of such images.

Fig. 3.30 Interpolation frame examples from the Middlebury dataset. [6].
Left: Frame-10 of 'Basketball' sequence an indoor scene. Right: Frame-10 of 'Dumptruck' sequence an outdoor image with some urban structures scene.

- Stereo data images: The last type of images are modified stereo images. The ground truth for such images were obtained using structured lighting. The displacement between images in stereo data are all horizontal, this makes estimation of the displacement field (disparity in case of stereo images) easier and more accurate. Figure-3.31 demonstrates the two frames of the 'Teddy' stereo sequence.

Fig. 3.31 Modified Stereo data examples from the Middlebury dataset. [6]. Left: Frame-10 of 'Teddy' dataset. Right: Frame-11 of 'Teddy' dataset.

MPI-Sintel

The MPI-Sintel dataset [150] is a synthetic image sequence taken from an animated 3D short film, it contains complex motion with varied textures. The ground truth was obtained with a complicated process that include identifying boundaries (for objects, materials, and depth), producing an estimate of the motion boundaries and thresholding with threshold gradient magnitude. The dataset is divided into two categories. The first is the training image category, which includes images with open-access ground truth. The second category is the test image category, with withheld ground truth. MPI-Sintel provides 1064 frames for training and 564 for testing. The frames were taken from 35 clips selected from the film.

Images used in this dataset are rendered in different levels, these levels are called 'passes'. The first level is 'albedo' which is the simplest rendering which does not contain illumination effect and has a piecewise constant colour. This means that the data (brightness) constancy assumptions holds across the whole image. The second level is the 'clean' rendering level which includes illumination effects (e.g. shading, specular reflections). The final level is the one that matches the final version of the film, which includes more complex

effects and adds motion blur, atmospheric effect, colour correction,etc.

Figure-3.32 includes an example of the MPI-sintel dataset. The example include the first frame of the image group 'alley-1', illustrating the three rendering passes. The figure also includes the ground truth obtained between that frame and the next frame in the image group. The ground truth follows the colour code in Figure-1.4.

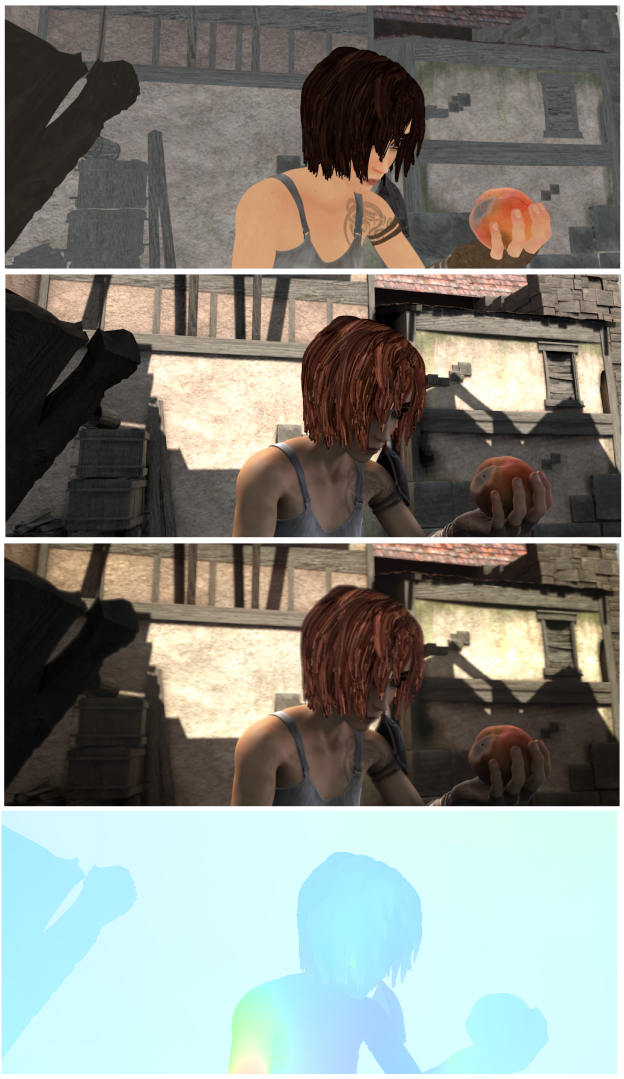


Fig. 3.32 Example of MPI-Sintel dataset [150].
This is frame-0001 of 'alley-1'.
Top: frame-0001 with albedo rendering. Second from top: frame-0001 with clean rendering. Third from top: frame-0001 with final rendering. Bottom: ground truth.

The KITTI dataset

Another dataset is the KITTI dataset [151], which is a multi-purpose dataset that can be used as a benchmark for optical flow, stereo matching, object detection, etc. The data was obtained using a complicated system of video cameras, laser scanners, and GPS/IMU units all mounted on a vehicle. The benchmark was aimed at autonomous driving and navigation. The ground truth for optical flow was obtained by registering images via point clouds using ICP (Iterative Closest Point), then calculating the optical flow by projecting 3D points into the sequence of registered images. However, they obtained a 50% density ground truth only, but it offered a benchmark for real world images, rather than images taken in a controlled lab environment.

3.4.2 Error Measures

There were several error measurements proposed to evaluate the performance of optical flow algorithms, perhaps the most popular one is the Angular Error (AE) and the End-point Error (EPE) [20] [5] [6]. AE finds the deviation of the computed optical flow in relation to the ground truth. This is done by calculating the angle between the vector of the computed optical field $\mathbf{w} = (u, v, 1)$ and the given ground truth flow field $\mathbf{w}_{gt} = (u_{gt}, v_{gt}, 1)$. A spatio-temporal flow vector is used preventing the division by zero at locations with no flow.

$$AE = arccos\Big(\frac{u \times u_{gt} + v \times v_{gt} + 1}{\sqrt{u^2 + v^2 + 1}\,\sqrt{u_{gt}^2 + v_{gt}^2 + 1}}\Big) \tag{3.23}$$

However this error measurement does not penalise all errors in a uniform way, as errors with large displacement values are penalised less severely than errors with small displacements [6].

End-point Error (EPE) is another error measurement [152], [6], it calculates the error as the square root of the sum of squared differenced between the computed and ground truth displacement fields:

$$EPE = \sqrt{(u - u_{gt})^2 + (v - v_{gt})^2} \tag{3.24}$$

Reconstruction error was also used to evaluate the efficiency of optical flow algorithms. Given two images in a sequence and the flow field computed between those images, it is possible to reconstruct one of the images using the other image and the computed optical

flow via interpolation. Lin and Barron in [153] compared the efficiency of several optical flow methods, the comparison was made depending on how well the reconstructed image compared to the actual image. Moreover several interpolation methods were investigated. The comparison was done by calculating the *Interpolation Error* (IE), which is the root-mean-square (RMS) between the reconstructed image and the actual second image.

$$IE = \sqrt{\frac{1}{N}\sum(I(x,y) - I_{GT}(x,y))^2} \tag{3.25}$$

where N is the number of pixels, I_{GT} is the actual second image, and I is the reconstructed image. An interesting finding of this paper was that the interpolation error correlates to the angular error in most cases, and hence interpolation error can be regarded as a good indicator for angular error. This is very useful especially in the case where no ground truth is available, as ground truth is very difficult to obtain.

Szeliski in [154] proposed a method to evaluate performance of optical flow algorithms without the need for ground truth data. The approach relies on assuming constant velocity for motion in images. Given a sequence of more than two images, the displacement field is calculated for a subset of two images, the flow field is then extrapolated to predict the third frame, or interpolated to construct an image lying between two frames. The quality of the algorithm is then judged by how efficiently it can reconstruct the predicted frame. The error calculated here is referred to as *'Interpolation Error'* (IE), which is the root-mean-square (RMS) difference between grey levels of the ground truth frame and the predicted interpolated (or extrapolated) frame (Equation-3.25). This type of error measurement was reported in the Middlebury database [6], along with the *'Normalised Interpolation Error'* (NE), which is given by:

$$NE = \sqrt{\frac{1}{N}\sum \frac{(I(x,y) - I_{GT}(x,y))^2}{|\nabla I_{GT}(x,y)|^2 + \varepsilon}} \tag{3.26}$$

where ε is an arbitrary scaling constant preventing division by zero.

3.5 Local and Global Optical Flow Methods

In general, variational methods for finding optical flow can be divided into local and global methods. The global methods for optical flow can generally produce dense flow fields, while the local methods are often more robust under the influence of noise. The work of

Horn-Schunk [1], and Lucas-Kanade [10] can be considered landmark papers for the global and the local methods receptively. In this section a brief introduction for both methods is presented. To reiterate on the notation that is used for the rest of this thesis, consider a rectangular image domain Ω with pixel coordinates (x,y). Optical flow algorithms aim at finding the displacement field $\mathbf{u} = (u,v)$ for corresponding pixels between images in an image sequence I_1 and I_2, which are taken at time t and $t+1$ respectively, u here is the displacement in the x direction, and v is the displacement in the y direction.

Variational methods usually start from the assumption that grey value levels (brightness) do not change over time, in what is called the data constancy assumption (Equation-2.3). The function resulting from this assumption is not convex, hence this function is linearised using a Taylor expansion, resulting in what is known as optical flow constraint (Equation-2.4). This function is ill-posed, and it is not possible to find a unique solution as this is one equation with two unknowns. This problem is referred to as the *'aperture problem'* [1], [12]. However, it is possible to find the flow parallel to image gradients (normal to image edges) [12]:

$$u_{parallel} = -\frac{I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|} \tag{3.27}$$

To overcome the ill-posedness, Lucas-Kanade [10] assumed that the flow is constant within a certain window (neighbourhood). The flow field is then found by least square minimisation of the following equation:

$$E_{LK} = K_\rho * [I_x u + I_y v + I_t]^2 \tag{3.28}$$

where u , and v are the flow fields, which are assumed constant for a window of size ρ centred at pixel (x,y). Subscripts denote partial derivatives. K_ρ is a Gaussian filter kernel of standard deviation ρ. The solution of the Lucas-Kanade equation is found by setting the derivatives with respect to u and v equal to 0, and the following set of equations are obtained:

$$K_\rho * \left[I_x^2\, u + I_x I_y\, v + I_x I_t \right] = 0 \tag{3.29}$$

$$K_\rho * \left[I_x I_y\, u + I_y^2\, v + I_y I_t \right] = 0 \tag{3.30}$$

Whether it is possible to find a solution or not depends on the size of the neighbourhood

and the available information in this neighbourhood. As the Lucas-Kanade method depends on image differentiation the information in a small neighbourhood may be insufficient to constrain the equation. As a solution for this issue it is possible to increase the neighbourhood size to include more information. However increasing the size of the neighbourhood means that it is more possible to include different motion in the same area, this renders the estimated flow field unreliable. Alternatively estimating the flow at sparse locations may be a good solution, the resulting flow field is not dense (sparse) in this case.

In a different approach, Horn-Schunck [1] proposed to add another constraint to the optical flow constraint (Equation-2.4). They assumed that the flow field changes smoothly across the image, therefore adding a smoothness (Regularity) term to the energy function. The energy function becomes:

$$E_{HS} = \int_{\Omega} \left((I_x u + I_y v + I_t)^2 + \alpha |\nabla \mathbf{u}|^2 \right) \, dxdy \tag{3.31}$$

where α here is a weight factor. $\nabla \mathbf{u} = (\nabla u, \nabla v)$. The basic assumption in the regularity term is a smooth flow field, and neighbouring pixels are expected to have similar displacements.

The solution is found via variational methods by obtaining the corresponding Euler-Lagrange equations. Euler-Lagrange are a set of partial differential equations, with a solution which is the minimisation of an energy function. The minimiser (u, v) of a function in the form [94]:

$$E(u,v) = \int_{\Omega} G(x, y, u, v, \nabla u, \nabla v) dxdy \tag{3.32}$$

Satisfying the following Euler-Lagrange set of equations:

$$\frac{\partial \, G_{u_x}}{\partial x} + \frac{\partial \, G_{u_y}}{\partial y} - \frac{\partial G}{\partial u} = 0 \tag{3.33}$$

$$\frac{\partial \, G_{v_x}}{\partial x} + \frac{\partial \, G_{v_y}}{\partial y} - \frac{\partial G}{\partial v} = 0 \tag{3.34}$$

Hence, the Euler-Lagrange equations for the Horn-Schunck equation (Equation-3.31) is given as follows:

$$\alpha\nabla^2 u - Ix(I_x u - I_y v - I_t) = 0 \tag{3.35}$$
$$\alpha\nabla^2 v - Iy(I_x u - I_y v - I_t) = 0 \tag{3.36}$$

where (∇^2) denotes the Laplace operator. This formulation allows the displacement fields u and v to be found at locations where image gradients approach 0. These displacements are induced from neighbouring pixels as an effect of the smoothness term, filling pixels with missing information. Hence, the resulting displacement flow calculated here is dense. Figure-3.33 next demonstrates visualisation for optical flow calculated via both the local method of Lucas-Kanade[2] [10] and the global one by Horn-Schunck [1].

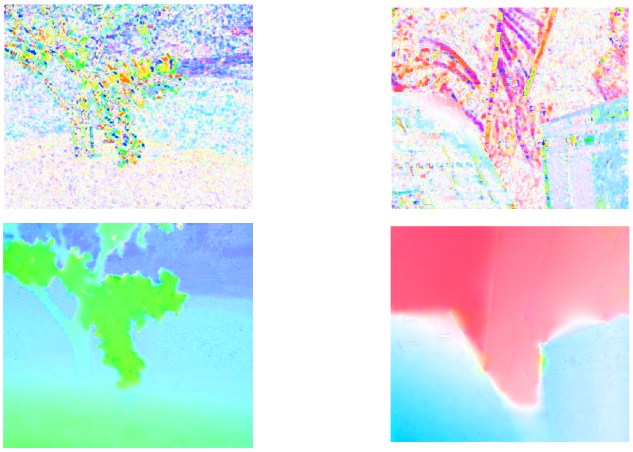


Fig. 3.33 Lucas-Kanade vs. Horn-Schunck optical flow computation.
Upper row: Optical flow field obtained via Lucas- Kanade local method. Lower row: Optical flow obtained via Horn-Schunck global method.

3.6 Combined Local-Global (CLG) Optical Flow

Lucas-Kanade energy function to estimate optical flow can be written in the following form [12]:

$$E_{LK} = \mathbf{w}^{\mathsf{T}} J_\rho(\nabla_3 I)\mathbf{w} \tag{3.37}$$

where $\nabla_3 I = (I_x, I_y, I_t)$, $J_0(\nabla_3 I) = K_\rho * (\nabla_3 I)$, K_ρ here is a Gaussian kernel filter with stan-

[2] MATLAB code was downloaded from http://crcv.ucf.edu/source/optical

dard deviation of ρ, and $\mathbf{w} = (u, v, 1)$. The Horn-Schunck energy function can also be re-written in the following form:

$$E_{HS} = \int_{\Omega} [\mathbf{w}^{\mathsf{T}} J_0(\nabla_3 I)\mathbf{w} + \alpha|\nabla\mathbf{w}|^2]\, dxdy. \tag{3.38}$$

The difference between Equation-3.31 and this equation is the spatio-temporal form of the latter one. Smoothing images plays an important role in improving the convergence of the solution. When an image is smoothed using a Gaussian kernel, the low-pass filter effect removes a certain percentage of the noise. However, this will also lead to loosing some of the small details in the image. In addition to that, when a certain window of pixels is smoothed, pixel values are changed as a function of the neighbourhood pixels values. In this way neighbouring pixels have a weighted effect on the calculation at a certain pixel. Hence, the calculation of flow field at a certain point is extended to include neighbouring pixels, and the estimated flow field at this point becomes more robust.

As was pointed out in the previous section, global methods for optical flow yield dense flow fields, while local methods are more robust. Hence Bruhn et al. [12] in suggested to combine the two methods to incorporate the effect of the local method into the global method of Horn-Schunck, Bruhn et al. [12] suggested to minimise the following equation

$$E_{CLG} = \int_{\Omega} [\Psi(\mathbf{w}^{\mathsf{T}} J_\rho(\nabla_3 I)\mathbf{w}) + \alpha\Psi(|\nabla_3\mathbf{w}|^2)]\, dxdy \tag{3.39}$$

where $\rho > 0$, $\Psi(.)$ is a robust penaliser, and $\nabla_3\mathbf{w}$ is a spatio-temporal smoothness term. The spatial filtering applied here is actually increasing the robustness of the data term by including the neighbourhood of the pixel to be part of the data constancy assumption. Meanwhile the estimated flow field resulting is also dense. The minimisation can be accomplished by obtaining the corresponding Euler-Lagrange equations:

$$0 = \sum_{j\in\mathcal{N}(i)} \frac{\Psi'_{2i} + \Psi'_{2j}}{2}(u_j - u_i) - \frac{\Psi'_{1i}}{\alpha}(J_{11i}u_i + J_{12i}v_i + J_{13i}) \tag{3.40}$$

$$0 = \sum_{j\in\mathcal{N}(i)} \frac{\Psi'_{2i} + \Psi'_{2j}}{2}(v_j - v_i) - \frac{\Psi'_{1i}}{\alpha}(J_{11i}u_i + J_{12i}v_i + J_{23i}) \tag{3.41}$$

Here $\Psi'_{1i} = \Psi'_{1i}(\mathbf{w}_i^{\mathsf{T}} J_{\rho i}\mathbf{w}_i)$, $\Psi'_{2i} = \Psi'_{2i}(|\nabla_3\mathbf{w}_i|^2)$, and i denotes a certain pixel in the image. J_{nm} is the structure tensor $J_\rho(\nabla_3 I)$. The solution for the simultaneous equations in 3.40 is

non-trivial, as they are non-linear. To solve such a system of equations, nested fixed-point iteration is used. In the inner loop Ψ' is kept fixed and the equations are solved to determine u and v. Several numerical schemes can be employed for this purpose such as Successive-Over Relaxation (SOR) [155]. The values obtained for (u,v) are then used to update Ψ'. The SOR iteration step was given in [12] as follows:

$$u_i^{k+1} = (1-\omega)u_i^k + \omega \frac{\sum_{j\in\mathcal{N}^-(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} u_j^{k+1} + \sum_{j\in\mathcal{N}^+(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} u_j^k - \Psi'_{1i}\frac{h^2}{\alpha}(J_{12i}v_i^k + J_{13i})}{\sum_{j\in\mathcal{N}^(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} + \Psi'_{1i}\frac{h^2}{\alpha}J_{11i}} \tag{3.42}$$

$$v_i^{k+1} = (1-\omega)v_i^k + \omega \frac{\sum_{j\in\mathcal{N}^-(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} v_j^{k+1} + \sum_{j\in\mathcal{N}^+(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} v_j^k - \Psi'_{1i}\frac{h^2}{\alpha}(J_{21i}u_i^{k+1} + J_{23i})}{\sum_{j\in\mathcal{N}^(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} + \Psi'_{1i}\frac{h^2}{\alpha}J_{22i}} \tag{3.43}$$

3.7 Preliminary Results

The main aim of the Preliminary experiments was to compare the local and global optical flow methods. These experiments were carried out at an early stage of the research [3]. A further experiment and investigation is presented in Section-5.1 of Chapter-5. The CLG optical flow was implemented in MATLAB, further details on the final implementation can be found in Section-5.1.1. The collection of images used here are taken from a video sequence with a frame rate of 25 frame/second [4]. Figure-3.34 shows two images taken from this sequence. Those two images were used as input to the local, global and the CLG optical flow algorithms. In these preliminary experiments the performance of the three algorithms was qualitatively assessed by investigating the flow field visualisation.

[3] These preliminary results were published in a paper and can be found in Appendix-B

[4] Video source: http://datasetfor.org/

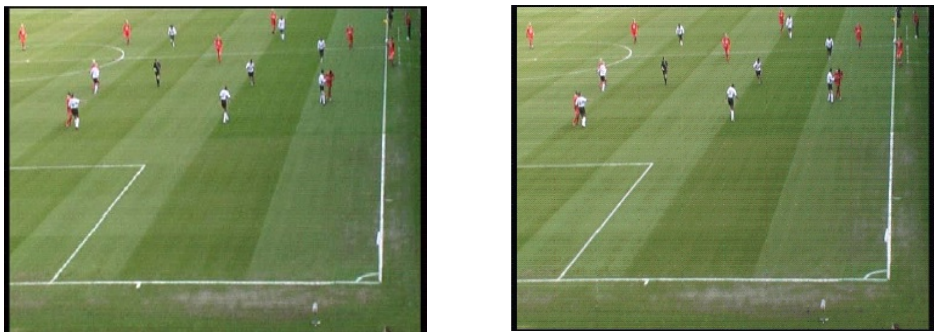


Fig. 3.34 FootballMatch image sequence.
Left: Image taken at time t. Right: Image taken at time t+1.

In Figure-3.35 the visualisation of the output from the local method is compared with the flow field of the CLG algorithm. By analysing both flow fields qualitatively, it can be seen that the overall view in the case of the CLG optical flow has better quality than that for the global flow case. Where the displacement field estimated using the CLG method is smoother and less affected by noise. Again all this analysis is done qualitatively, further quantitative results are shown later in Chapter-5. On the other hand the local method does not result in a dense flow field. In general, the optical flow field in the CLG case is more obvious around the visible motion field, while in global methods the results are more noisy due to the nature of such algorithms. As for the local method, and as pointed out earlier, the flow field is not dense. A further investigation is done by showing a zoomed-in area of the results in order to clearly analyse the results.

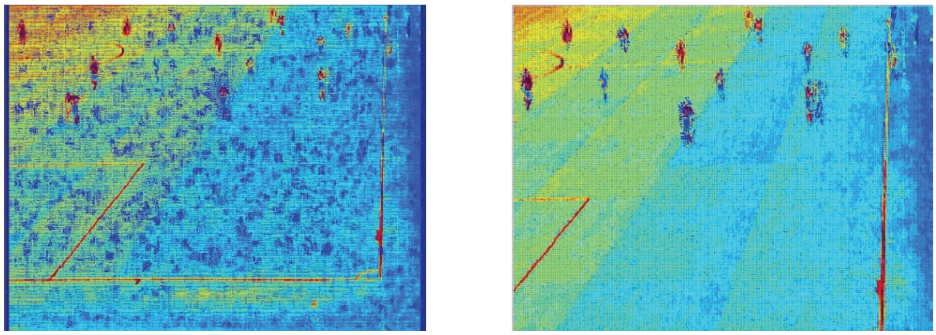


Fig. 3.35 A comparison of the optical flow field computed in the global and CLG cases.
Left: Optical flow field obtained using Global method. Right: Optical flow field obtained using CLG method.

In Figure-3.36 next a comparison of zoomed-in versions of the estimated flow field using the three methods.

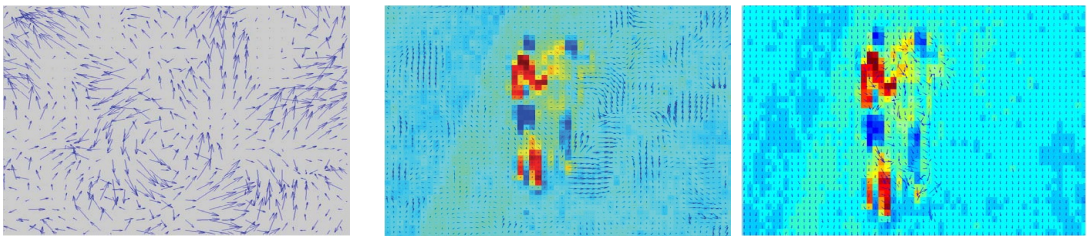


Fig. 3.36 Optical flow zoom-in comparison.
Left: Zoomed-in local flow field. Middle: Zoomed-in global flow field. Right: Zoomed-in CLG flow field.

3.8 Improving Edge-preserving in CLG Optical Flow

The initial results in the previous section were obtained using the original CLG optical flow method [12]. The data term used in the CLG method uses a smoothed version of the intensity values of the image, which results in a more robust calculation. However, the use of Gaussian smoothing in the data terms introduces blurriness which becomes apparent at the edges. In order to improve the quality of the estimated displacement fields, and to improve edge-preserving performance, further experiments are conducted using an improved version of the CLG. In the proposed improved version linear filtering is replaced with a non-linear edge preserving filter, that is the bi-lateral filter [60], [59]. The use of the bi-lateral filter here is similar to that proposed in [4], however in the current case it is applied to the CLG algorithm [12]. In this section a formulation of an improved CLG optical flow method is presented, this version is based on the used of bi-lateral filtering to obtain an edge-preserving structure tensor. Experiments are presented in Chapter-5 to illustrate the improvement that this method gives in image registration over the global method of Horn-Schunck [1].

The bilateral filter was discussed in Section-3.2.2, the kernel weights for this filter varies according to the spatial distance from the pixel and also according to pixel value similarity. To improve the CLG performance, it is proposed to replace the Gaussian filter in Equation-3.38 with the bi-lateral filter kernel, hence the equation becomes:

$$E_{HS} = \int_{\Omega} [\mathbf{w}^{\mathsf{T}} Jbf(\nabla_3 I)\mathbf{w} + \alpha|\nabla \mathbf{w}|^2]\, dxdy \tag{3.44}$$

where $Jbf(\nabla_3 I)$ is the structure tensor obtained using the bilateral filter kernel [4], and can be written following Equation-3.20 as follows:

$$Jbf = kbf * J_0.$$

The derivation of the Euler-Lagrange equations is similar to the Gaussian CLG optical flow, which results the following sets of equations:

$$0 = \sum_{j \in \mathcal{N}(i)} \frac{\Psi'_{2i} + \Psi'_{2j}}{2}(u_j - u_i) - \frac{\Psi'_{1i}}{\alpha}(Jbf_{11i}u_i + Jbf_{12i}v_i + Jbf_{13i}) \tag{3.45}$$

$$0 = \sum_{j \in \mathcal{N}(i)} \frac{\Psi'_{2i} + \Psi'_{2j}}{2}(v_j - v_i) - \frac{\Psi'_{1i}}{\alpha}(Jbf_{11i}u_i + Jbf_{12i}v_i + Jbf_{23i}) \tag{3.46}$$

The non-linearity in this system of equations makes the solution difficult. However this can be solved by applying nested fixed-point iterations similar to Equation-3.42, and Equation-3.43. The outer iteration updates Ψ', while the inner iteration updates the values of u, and v.

$$u_i^{k+1} = (1-\omega)u_i^k + \omega \frac{\sum_{j \in \mathcal{N}^-(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} u_j^{k+1} + \sum_{j \in \mathcal{N}^+(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} u_j^k - \Psi'_{1i}\frac{h^2}{\alpha}(Jbf_{12i}v_i^k + Jbf_{13i})}{\sum_{j \in \mathcal{N}(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} + \Psi'_{1i}\frac{h^2}{\alpha}Jbf_{11i}} \tag{3.47}$$

$$v_i^{k+1} = (1-\omega)v_i^k + \omega \frac{\sum_{j \in \mathcal{N}^-(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} v_j^{k+1} + \sum_{j \in \mathcal{N}^+(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} v_j^k - \Psi'_{1i}\frac{h^2}{\alpha}(Jbf_{21i}u_i^{k+1} + Jbf_{23i})}{\sum_{j \in \mathcal{N}(i)} \frac{\Psi'_{2i}+\Psi'_{2j}}{2} + \Psi'_{1i}\frac{h^2}{\alpha}Jbf_{22i}} \tag{3.48}$$

3.9 Summary

In this chapter the relation between image registration and optical flow was explored. A CLG optical flow method for displacement field estimation was used for image registration. In addition to that an improved CLG method is proposed which replaces the Gaussian filter-

ing in the data term with an edge preserving bi-lateral filter. The chapter included also the background theory necessary to implement any optical flow algorithms. This includes some basic and fundamental concepts such as filtering and image resizing, in addition to some more complex notions such as structure tensor.

Some initial results are reported in this chapter, including some preliminary comparison for image registration using the local, global and CLG optical flow. Further experiments are reported in Chapter-5. These experiments are designed to highlight the improved performance of the bi-lateral CLG optical flow and compare the results with its counterpart the global optical flow proposed by Horn-Schunk [1].

Chapter 4

Steered-L^1 Norm for Optical Flow Calculation

There are two main components that can be identified in the energy function used to estimate optical flow (Equation-2.1). The first is the data term which is based on the assumption of unchanged illumination. The second is the smoothness or the regularity term which is based on the assumption that the flow field is smooth or piecewise smooth in nature. More specifically the flow field is piecewise smooth, therefore the smoothness term should be chosen carefully to characterise this property of the flow field. Several smoothness terms can be used in the computation of optical flow, among those is the piecewise robust L^1 norm. Despite its plausible characteristics, the L^1 norm is not without its problems. The L^1 norm is not continuously differentiable, this issue was addressed in the Total-Variation dual algorithm $TV-L^1$ presented by Zach et al. [22].

One of the main advantages of the inclusion of a smoothness term in global variational optical flow is the filling-in effect that such algorithms demonstrate (see Section-1.3, 2.2). Unlike the local method, a flow field can be estimated at a certain point in global methods even if no information is available at that point of the image. In this case the flow field estimation is induced from the surrounding neighbourhood information available there. The L^1 norm provides a good filling-in performance. In addition to that the L^1 norm has an excellent performance in terms of preserving edges. The penalisation of this norm allows for discontinuities in the flow field. However, the filling-in effect decreases near edges as the penalisation decreases. This happens isotropically, in other words the filling-in effect decreases along motion boundaries.

In this chapter, an improved L^1 smoothness term is proposed that can be considered anisotropic

in terms of penalisation near image edges. The L^1 is steered according to local image structures. The direction of the regularity term is adapted to local image structure, while the penalisation magnitude is adapted to the flow field itself. The regularity is decomposed into components orthogonal and along image edges, hence encouraging the filling-in effect [15], [124], [11].

This chapter includes the following main sections: a section of introduction to dual $TV-L^1$ optical flow algorithm [22]. A section discussing the influence of different regularity terms on the calculation of optical flow; a section introducing the proposed steered-L^1 norm regularity term; a section on the derivation of a robust data term to work in the dual formulation minimisation, which is inspired by the work of Brox et al. [2].

4.1 Total Variation Optical Flow

The $TV-L^1$ smoothness term can handle discontinuities better than the L^2 norm used by Horn-Schunck [1]. However, this term is not continuously differentiable. Zach et al. [22] proposed a dual formulation to calculate optical flow based on the numerical scheme presented by Chambolle [91] that was proposed to solve the ROF total variation based image denoising [63]. The optimisation problem is split into two optimisation steps using an auxiliary variable $\mathbf{z}$ which is a close approximation for the displacement field $\mathbf{u}$. Starting from the non-linearised data constancy term, the energy function can be written as:

$$E = \int_{\Omega} (|I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})| + |\nabla \mathbf{u}|) d\mathbf{x} \tag{4.1}$$

I_2 is then linearised:

$$I_2(\mathbf{x}+\mathbf{u}) = I_2(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_2(\mathbf{x}+\mathbf{u_0}) \tag{4.2}$$

where $\mathbf{u}_0$ is the displacement initialisation. In the C2F framework $\mathbf{u}_0$ is the propagated displacement from the previous pyramid layer, and it has a value of $(0,0)$ in the first pyramid layer. In the latter layers, the value of $\mathbf{u}_0$ is the initial value of the displacement field that was computed in the previous layer of the pyramid (up-scaled to the current layer). To further illustrate $\mathbf{u}_0$, let $\delta\mathbf{u}$ be the displacement computed at a certain layer of the C2F pyramid, then the total displacement can be expressed as $(\mathbf{u} = \mathbf{u}_0 + \delta\mathbf{u})$, in other words the displacement

is equal to the displacements computed in the previous pyramid layer plus the displacement computed at the current layer. Equation-4.1 becomes:

$$E = \int_{\Omega} (\, |\mathbf{u}\, I_{2\mathbf{x}} + I_2(\mathbf{x}+\mathbf{u}_0) - \mathbf{u}_0\, I_{2\mathbf{x}} - I_1| + |\nabla \mathbf{u}| \,) d\mathbf{x} \tag{4.3}$$

where $I_{2\mathbf{x}} = (I_{2x},\ I_{2y})$ are the image gradients in the x, and y directions. The auxiliary variable ($\mathbf{z} = (zx,\, zy)$) is introduced at this point:

$$E = \int_{\Omega} (\overbrace{\alpha|\rho(\mathbf{z})| + \underbrace{\frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2}_{E_{dual}} + |\nabla \mathbf{u}|}^{E_{primal}} \,) d\mathbf{x} \tag{4.4}$$

where θ is a small constant, and α is the weight constant of the data term, this weight is manually tuned to give the best performance (A process similar to the one found in 5.2.2). This Equation is similar to Equation-4.1 but with the added term $\frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2$, which is added to decouple the minimisation problem into two sub-problems. The terms $I_2(\mathbf{x}+\mathbf{u_0}) - (\mathbf{u} - \mathbf{u_0})I_{2\mathbf{x}} - I_1$ in Equation-4.3 is denoted as the residual $\rho(\mathbf{u})$ in the current equation. An alternating primal-dual formulation is adopted to minimise the previous energy function. In the dual step:

$$E_{dual} = \int_{\Omega} (\, |\nabla \mathbf{u}| + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \,) d\mathbf{x} \tag{4.5}$$

where this is the dual problem obtained from Equation-4.4 as indicated using the braces. The aim of the minimisation of this equation if to find the minimum u, while the auxiliary variable ($\mathbf{z} = (zx, zy)$) is fixed. The minimisation process can be done in several ways, such as obtaining the Euler-Lagrange equations, or by applying the divergence theorem[22], [4], [23], [91], which leads to the following equation in the x direction:

$$u = z_x - \theta \ \texttt{div}\ \mathbf{p}_u \tag{4.6}$$

and similarly for the displacement in the y direction:

$$v = z_y - \theta \ \texttt{div}\ \mathbf{p}_v \tag{4.7}$$

where $\mathbf{p}_u = (p_{1u},\ p_{2u})$ and $\mathbf{p}_v = (p_{1v},\ p_{2v})$.

The value of $\mathbf{p}_u$ is found as proposed Chambolle [91] using a semi implicit gradient descent algorithm as shown in the next equation (It is worth noting here that the derivation of the dual step is shown in more details in Section-4.3.1 as it is core part of the algorithm proposed in this thesis).

$$\mathbf{p}_u^{k+1} = \frac{\mathbf{p}_u^k + \tau.\nabla\Big(div(\mathbf{p}_u^k) + zx/\theta\Big)}{1 + \tau.\Big|\nabla\big(div(\mathbf{p}_u^k) + zx/\theta\big)\Big|} \tag{4.8}$$

where $\tau \leq 1/8$, and $p^0 = 0$. A similar equation can be obtained to find the minimisation in the other direction (i.e for $\mathbf{p}_v$).

In the primal step, the aim is to minimise the following function:

$$E_{primal} = \int_\Omega (\ \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 + \alpha|\rho(\mathbf{z})|\ )d\mathbf{x} \tag{4.9}$$

The minimisation for this problem is performed via a thresholding step:

$$\mathbf{z} = \mathbf{u} + \begin{cases} \alpha\theta I_\mathbf{x} & if : \rho(\mathbf{u}) < -\alpha\theta I_\mathbf{x}^2 \\ -\alpha\theta I_\mathbf{x} & if : \rho(\mathbf{u}) > \alpha\theta I_\mathbf{x}^2 \\ -\rho(\mathbf{u})/I_\mathbf{x} & if : |\rho(\mathbf{u})| \leq \alpha\theta I_\mathbf{x}^2 \end{cases} \tag{4.10}$$

The aim of this thresholding step is to minimise the residual:

$$\rho = \mathbf{w}I_\mathbf{x} + I_t \tag{4.11}$$

This is done by adding or subtracting displacements proportional to the image gradients. Hence, $\mathbf{z}$ makes a leap if the magnitude of the residual ρ is bigger than a certain threshold, and makes a small step if the residual is small. In this way the residual is allowed to vanish gradually. One would expect that if wrong initialisation was used, the displacement could converge to wrong minima and residuals would not be minimised.

4.2 Regularisation Influence on Optical Flow Performance

As discussed earlier, the smoothness term is a very important part of a variational optical flow energy function. It was introduced based on the assumption that a flow field usually

exhibits a smooth displacements, however this assumption does not always hold. Since the optical flow field of a certain scene includes a collection of several motions, the resulting displacement field is piecewise smooth [66]. The marquee work of Horn-Schunck [1] used quadratic function $\Psi(s) = s^2$ (see Equation-2.12). This regularisation term severely penalises diverse displacements of neighbouring pixels including the areas with motion discontinuities. The effect of such penalisation is a blur across these discontinuities (see Figure-1.8). Since the quadratic function penalises the flow field equally in all directions in the same magnitude, it can be referred to as *'homogeneous smoothness term'* [94]. Several improved versions were proposed to enhance the performance of the regularisation in the presence of a piecewise flow field. Ideally a piecewise function will suffice to characterise the piecewise smooth displacement field. Isotropic, anisotropic flow and image driven regularisations were proposed by many researchers to improve the performance of smoothness terms with regard to motion boundaries conservation. *'Robust Statistics'* were used in optical flow calculations to improve the performance of the estimated field, in particular robust statistical functions (estimators) were used in the regularisation terms to render these terms less sensitive to outliers [64].

Robust statistics are concerned with the problem of estimation in the presence of outliers [156]. In the context of optical flow estimation, motion boundaries can be considered as outliers present between two smooth displacement regions [157]. Several robust functions with different performance were used to calculate a flow-driven optical flow field [13], [12], [66], [2]. In this section a brief comparison between different regularisation terms is presented. The discussed terms include the quadratic and first norm, in addition to several robust estimators. These smoothness terms are juxtaposed and compared in terms of their *Influence Function*. The influence function is proportional to the first derivative of the robust function [157], [156], it helps to analyse the performance of such estimators and shows the behaviour in response to a certain measurement [157].

Consider the quadratic function $\Psi(s) = s^2$, with the derivative $\Psi'(s) = 2s$. Figure-4.1 depicts the function with its derivative. By examining the figure, it can be seen that the influence value increases linearly with the increase of s without bound [157]. In the context of optical flow, this can be interpreted as follows: if a pixel exists near a motion boundary, the displacement field gradient increases ($s = \nabla \mathbf{u}$), and since there is no limit for the influence, the value of estimation will be affected by the motion field across these motion edges. Hence, blurred motion boundaries are created.

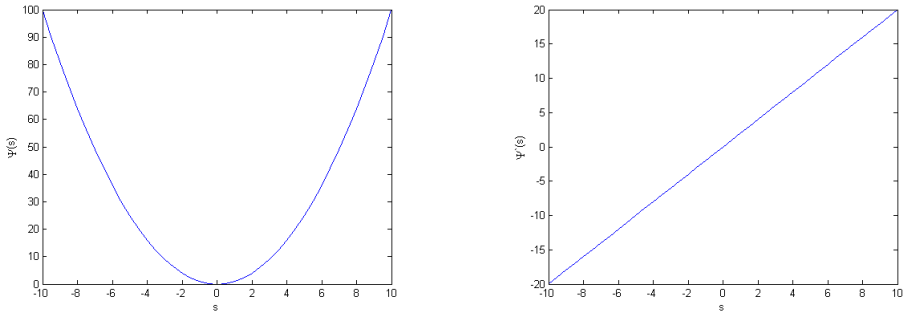


Fig. 4.1 Quadratic estimator and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s)$.

The Lorentzian robust function given in Equation-2.16 was also used in several optical flow algorithms [64] [65] [66]. Its influence function (derivative) is given as follows:

$$\Psi'(s) = \frac{2s}{2\varepsilon^2 + s^2}. \tag{4.12}$$

Figure-4.2 demonstrates the estimator and its influence function.

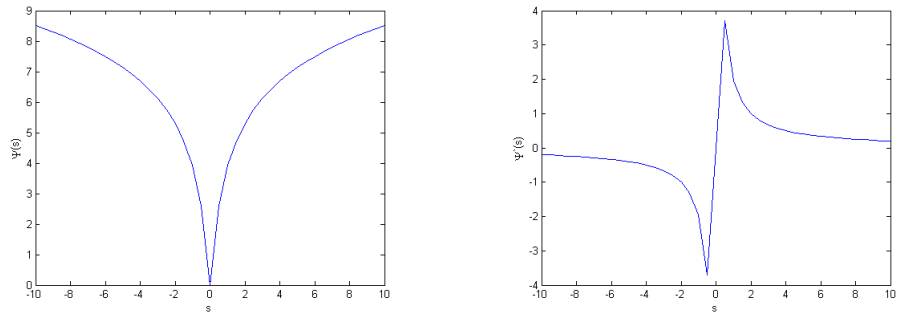


Fig. 4.2 Lorentzian estimator and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s^2)$.

The Lorentzian estimator is more robust than the quadratic function, as can be seen from the figure. As the s value increases the influence function increases up to a certain level where it starts decreasing. The threshold point where the function's value starts decreasing is determined by the value of the constant ε. In the case of optical flow this can be interpreted as follows: the displacement field will be penalised increasingly as the gradients of the flow field increases. At a certain threshold the penalisation starts decreasing which makes the smoothing affect less severe in the calculation of the flow field. Hence, motion edges are preserved more efficiently. Similarly for the Charbonnier estimator (see Equation-2.15),

which has the following derivative [12]:

$$\Psi'(s^2) = \frac{1}{\sqrt{1+\frac{s^2}{\varepsilon^2}}}. \tag{4.13}$$

Figure-4.3 next depicts the function with its derivative:

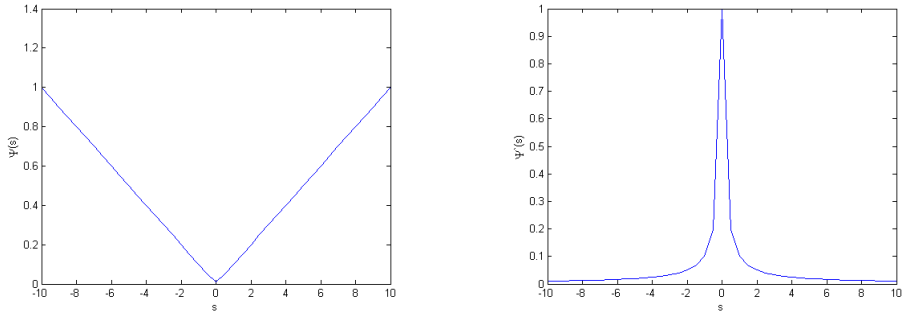


Fig. 4.3 Charbonnier estimator and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s)$.

The Charbonnier penalisation degrades gradually, however it does not reach zero.

The L^1 norm (Equation-4.14, 4.15) is another function that can be used as a smoothness term of the optical flow energy function. It provides a robust penalisation, and the piecewise nature of this function seems plausible for that purpose:

$$\Psi(s) = |x| \tag{4.14}$$

$$\Psi'(s) = \frac{x}{|x|}. \tag{4.15}$$

Figure-4.4 next depicts the L_1 norm and its derivative:

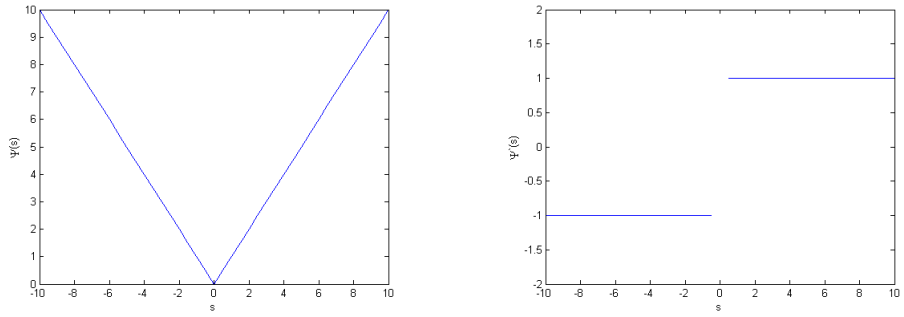


Fig. 4.4 $L1$-norm and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s)$.

However, as discussed earlier, it is obvious from Figure-4.4 that the L^1 norm is not continuously differentiable.

The L^1 norm can be approximated via the following function:

$$\Psi(s) = \sqrt{s^2 + \varepsilon^2} \tag{4.16}$$

where ε here is also a small constant. This constant helps to avoid dividing by zero in the derivative. A small value of ε results in a slow convergence of the solution, while a high value of this constant results a blurry boundaries. The influence of this robust estimator is given by:

$$\Psi'(s) = \frac{s}{\sqrt{s^2 + \varepsilon^2}} \tag{4.17}$$

and the influence function can be depicted as follows:

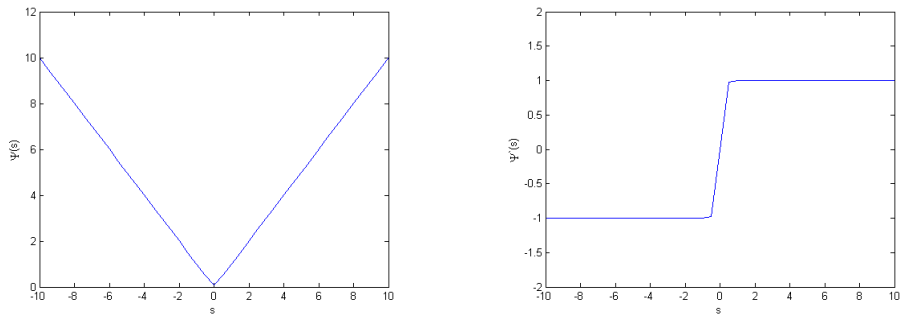


Fig. 4.5 L^1 norm approximation and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s)$.

The influence function of this estimator (Figure-4.5) has a similar behaviour and influence function for that of the L^1 norm. An important question may arise here, which is **why the L_1 norm is considered robust and can preserve motion edges in comparison to the quadratic norm ?** The answer to this question can be found by examining the influence function of the L^1 norm (Figure-4.4), where the piecewise nature of this function can characterise the piecewise motion flow field. Unlike the quadratic penalisation function, the influence of the increase of s in the L^1 norm does not increase with the increase of s but rather stays constant. This definitely means the L^1 norm is a better smoothness term in comparison with the quadratic penalisation where the influence increases without limits.

A modified version of the robust L^1 norm was also used and is illustrated in the following equations:

$$\Psi(s^2) = \sqrt{s^2 + \varepsilon^2} \tag{4.18}$$

$$\Psi'(s^2) = \frac{1}{2\sqrt{s^2 + \varepsilon^2}}. \tag{4.19}$$

This function offers a robust performance and was used in many optical flow algorithms [13], [2], [84], [61]. Figure-4.6 next depicts the function with its derivative:

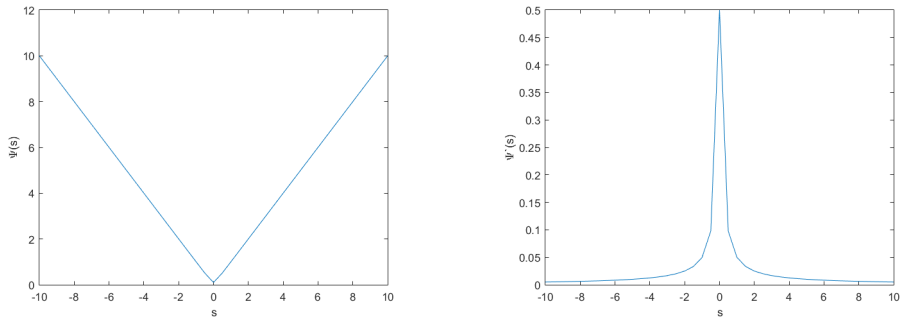


Fig. 4.6 Modified L^1-norm approximation and its influence function.
Left: $\Psi(s)$. Right: $\Psi'(s)$.

Ideally the smoothness term should descend to zero near edges to decrease the penalisation effect and to prevent the blurring at motion discontinuities. The Lorentzian and the Charbonnier estimators exhibit a good performance in this aspect where the influence decreases at a certain point (although does not descend to zero). However one issue with such an estimator (including the robust estimator in Equation-4.16) is their sensitivity to the constant ε.

Indeed these robust estimators (e.g. Equations-4.18) performance is related to the choice of the constant ε. A small value of ε would render the convergence of the function slow, while large values of ε results in a blur across the motion boundaries as properties of the model are lost [9], [22], [158]. Therefore the robust L^1 norm is chosen here as the smoothness term for the optical flow model. To overcome the problem of discontinuous differentiability of the L^1 norm, a dual minimisation framework was used in the optical flow algorithm presented in [22]. In addition to the previous advantages, the dual minimisation offers an easy implementation on modern graphic cards, enabling such algorithms to have a real or near real-time performance [38], [22]. Although the performance speed is outside the context of this thesis, it is useful to point out such a fact for future work if a real-time version of this algorithm is required. In the next section an steered-L^1 smoothness term is presented.

4.3 Steered $TV - L^1$ Regularisation for Optical Flow Calculations

In the previous section the reason for choosing the L^1 norm in the smoothness term was discussed. However, the L^1 norm suffers from several problems, some of them were pointed out in the previous section. One of the issues that degrades the performance of the L^1 norm is being isotropic, i.e. it penalises the flow field in all directions regardless of the local structure [4]. Hence, at regions near motion boundaries, the penalisation of this norm decreases in all directions. The decrease in penalisation allows for motion discontinuities at such motion boundaries in order to preserve these edges. At the same time this decrease in penalisation decreases the filling-in effect parallel (along) such boundaries. This eventually reduces the accuracy of the estimated flow field.

Several researchers tried to address this issue by adding a diffusion filter in the smoothness term. The reason for this addition is to reduce penalisation across image areas that correspond to image edges, and to strengthen this along these edges to encourage the filling-in effect [4], [23]. Several choices of diffusion filter D are available such as:

$$D(\nabla I) = e^{-(\frac{|\nabla I|}{\beta})^2}$$

or:

$$D(\nabla I) = \frac{1}{1+(\frac{|\nabla I|}{\beta})^2}$$

where β is a constant [93]. These diffusion filters are functions of the image gradients, therefore the penalisation influence will depend on image edges rather than the flow field edges. This produces over-segmentation in the estimated optical flow field. The reason for this is that the magnitude of penalisation is varying with the magnitude of the image gradient. At relatively smooth areas of the image, the value of ∇I is relatively small and hence $D(\nabla I)$ is high, which allows for strong penalisation. However, as the pixels approach an image edge, the opposite takes place and $D(\nabla I)$ decreases, encouraging over-segmentation in areas corresponding to image edges.

To overcome the over-segmentation issue it seems a good idea to vary the penalisation magnitude in accordance to the flow field variation itself. In this section a steered L^1 norm optical flow algorithm is presented, which resembles the work of an anisotropic regularity. The smoothness term is designed to preserve motion edges while encouraging the filling-in effect (along such edges) that the global optical flow algorithms demonstrate.

The concept of *'Steered image derivative'* was introduced earlier [159], [40]. The idea is to steer image derivatives from the conventional x, and y directions to directions orthogonal to, and aligned with local image structure. This can be formulated as:

$$I_o = \cos\phi.I_x + \sin\phi.I_y \tag{4.20}$$

$$I_a = -\sin\phi.I_x + \cos\phi.I_y \tag{4.21}$$

where ϕ is the angle of the first eigenvector and it is obtained via the structure tensor matrix. Equation-4.20, and Equation-4.21 can be written in the following compact form:

$$I_{\mathbf{s}} = \mathbf{e}^{\mathsf{T}} I_{\mathbf{x}} \tag{4.22}$$

where $I_{\mathbf{s}} \in \{I_o, I_a\}$ are the image derivatives orthogonal to I_o and aligned with I_a image structures, and $\mathbf{e} \in \{e_1, e_2\}$ are the two eigenvectors of the structure tensor. The image structure is obtained via the structure tensor discussed earlier in Section-3.3. Sun et al. [11] used this to calculate optical flow in the discrete setting, they combined the image and flow-driven in one smoothness term. Later, the idea was picked-up by Zimmer et al. [15], [67] and used this combination in their smoothness term to improve optical flow performance and edge conservation. In their paper they used robust estimators in the smoothness term.

Starting from the image structure tensor (see Equation-3.20), two orthonormal eigenvectors can be obtained from this matrix. The first eigenvector pointing orthogonal to the

image structure $(\cos\phi, \sin\phi)$, and the other along these structures $(-\sin\phi, \cos\phi)$. Figure-4.7 shows the two eigenvectors at a certain point near an image edge, for a structure tensor which was obtained using a 5×5 Gaussian filter. The first line plotted in green is the eigenvector corresponding to the highest eigenvalue and it is pointing orthogonally to the predominant image structure (edge). The second line plotted in red is the other eigenvector and it is aligned with the image structure.

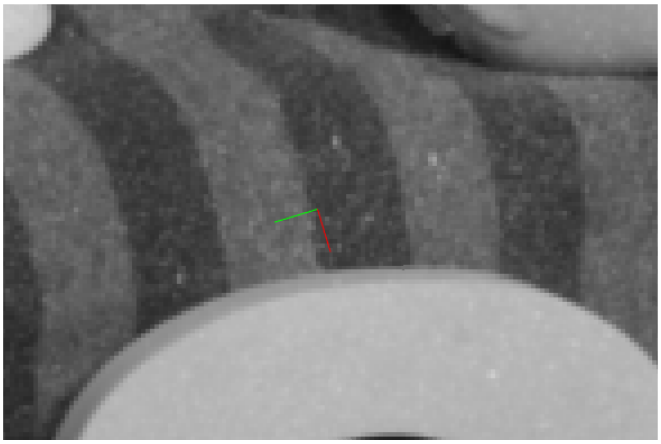


Fig. 4.7 Eigenvectors directions of a structure tensor.
The green line corresponds to the first eigenvector, while the red line corresponds to the second eigenvector.

The aim of the optical flow algorithm presented here is to minimise the following function:

$$E = \int_{\Omega} \Big(E_{data}(I_1, I_2) + E_{smooth}(\mathbf{u}, \nabla\mathbf{u}, I_1) \Big). \tag{4.23}$$

The smoothness term used in this algorithm is the L^1 norm which is not continuously differentiable. Therefore the minimisation follows the dual formulation that was proposed by Zach et al. [22] and discussed earlier in Section-4.1. To this end the auxiliary variable $\mathbf{z}$ is introduced, and the minimisation problem takes the following form:

$$E = \int_{\Omega} \Big(\alpha E_{data}(I_1, I_2) + \frac{1}{2\theta}(\mathbf{u} - \mathbf{z})^2 + E_{smooth}(\mathbf{u}, \nabla\mathbf{u}, I_1) \Big) \tag{4.24}$$

where $\mathbf{z}$ and θ are defined in Section-4.1, and α is the data term weight in this formula. The problem is broken down into two minimisation steps, a dual step (Equation-4.5), and a primal step (Equation-4.9). In the following subsection, a detailed discussion for minimisation of the data and smoothness terms for this equation is presented.

4.3.1 The Dual Step (Smoothness Term)

The aim of the dual step is to minimise **u** while keeping **z** fixed. The dual equation can be re-written in accordance with the flow directions u and v in the following form [11], [15]:

$$E_{dual} = \int_{\Omega} \Big(|\nabla u| + \frac{1}{2\theta}(u - zx)^2 + |\nabla v| + \frac{1}{2\theta}(v - zy)^2 \Big).$$

Here the steered $TV - L^1$ regularity term is presented. The two orthonormal eigenvectors of the structure tensor e_1 and e_2 are obtained. A modified version of the smoothness is proposed and the dual equation is written as follows [11], [15], [67]:

$$E_{dual} = \int_{\Omega} \Big(|\mathbf{e}^{\mathsf{T}} \nabla u| + \frac{1}{2\theta}(u - zx)^2 + |\mathbf{e}^{\mathsf{T}} \nabla v| + \frac{1}{2\theta}(v - zy)^2 \Big) \tag{4.25}$$

where:

$$\mathbf{e}^{\mathsf{T}} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix}.$$

In this way the directions of the smoothness term is adapted to the direction of the local image structure, while the magnitude of the penalisation if adapted to the flow field itself [15]. The minimisation of the dual step (Equation-4.25) is non-trivial, this is due to the non continuous differentiability of the L^1 norm.

To solve the minimisation, Euler-Lagrange equations are obtained (see Equation-3.33, 3.34). The first equation obtained is the following [4]:

$$-div\big(\mathbf{e}^{\mathsf{T}}.\frac{\nabla u}{|\nabla u|}\big) + \frac{1}{\theta}(u - zx) = 0 \tag{4.26}$$

Let $\mathbf{p}_u = \frac{\nabla u}{|\nabla u|}$, hence:

$$-div\big(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u\big) + \frac{1}{\theta}(u - zx) = 0 \tag{4.27}$$

which can be re-written as follows:

$$u = \theta.div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + zx \tag{4.28}$$

where $\mathbf{p}_u = (p_u^1, p_u^2)$. It follows that:

$$\mathbf{p}_u.|\nabla u| - \nabla u = 0, \qquad |\mathbf{p}_u| \leqslant 1 \tag{4.29}$$

Substituting Equation-4.28 in Equation-4.29, the following equation is obtained:

$$\mathbf{p}_u.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + zx/\theta\big)\Big| - \nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u) + zx/\theta\big) = 0. \tag{4.30}$$

Adding $\mathbf{p}_u$ to both sides of the above equation yields the following fixed-point iteration to find $\mathbf{p}_u$:

$$\mathbf{p}_u^{k+1} = \frac{\mathbf{p}_u^k + \tau.\nabla\Big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k) + zx/\theta\Big)}{1 + \tau.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k) + zx/\theta\big)\Big|} \tag{4.31}$$

where k is the iteration count, and τ is the step size. In the same way $\mathbf{p}_v$ can be obtained, and it is calculated using the following fixed-point iteration:

$$\mathbf{p}_v^{k+1} = \frac{\mathbf{p}_v^k + \tau.\nabla\Big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k) + zy/\theta\Big)}{1 + \tau.\Big|\nabla\big(div(\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k) + zy/\theta\big)\Big|}. \tag{4.32}$$

The terms $\mathbf{e}^{\mathsf{T}}.\mathbf{p}_u^k$, and $\mathbf{e}^{\mathsf{T}}.\mathbf{p}_v^k$ can be replaced by the alternative notations $\mathbf{p}_{su}$, and $\mathbf{p}_{sv}$, where:

$$\mathbf{p}_{su} = \mathbf{e}^{\mathsf{T}}.[p_{1u},\ p_{2u}]^{\mathsf{T}}$$
$$\mathbf{p}_{sv} = \mathbf{e}^{\mathsf{T}}.[p_{1v},\ p_{2v}]^{\mathsf{T}}$$

Hence, Equation-4.31, and Equation-4.32 can be written in the following way:

$$\mathbf{p}_u^{k+1} = \frac{\mathbf{p}_u^k + \tau.\nabla\Big(div\ \mathbf{p}_{su} + zx/\theta\Big)}{1 + \tau.\Big|\nabla\big(div\ \mathbf{p}_{su} + zx/\theta\big)\Big|}. \tag{4.33}$$

$$\mathbf{p}_v^{k+1} = \frac{\mathbf{p}_v^k + \tau.\nabla\Big(div\ \mathbf{p}_{sv} + zy/\theta\Big)}{1 + \tau.\Big|\nabla\big(div\ \mathbf{p}_{sv} + zy/\theta\big)\Big|}. \tag{4.34}$$

4.3.2 Data Fidelity Term

Several data terms were previously proposed in numerous algorithms. One of the first modules was the linearised data fidelity term of Horn-Shuck [1], which is based on the assumption of brightness constancy between images. However this assumption usually gets violated due to illumination changes. To cope with such changes several techniques can be followed. For example structure-texture decomposition can be used to mitigate the effect of illumination changes that are attributed to shading reflection and shadows [14], [4]. Alternatively image derivatives can be added to the data fidelity term to increase robustness against illumination changes [2]. In addition to incorporating the image derivative, the proposed data term of Brox et al. [2] postponed data term linearisation in order to enable the capturing of large displacements which offered high accuracy in the results. In the dual minimisation formulation, structure-texture decomposition was used in several algorithms while there is no attempt to use the image gradient in the data term.

As stated earlier, the focus of this thesis is on the performance of the proposed steered-L^1 norm smoothness term. Additionally a data term which incorporates image derivatives in the data fidelity term is used here. Although image derivatives were used before to improve robustness against illumination changes, its formulation was not derived in the dual minimisation framework before. Braux-Zin et al. [98] proposed a similar dual formulation to compute optical flow. However, an intrinsic difference is that they use AD-Census in the data fidelity term [99]. In this subsection a derivation for a data term is presented, this term includes the data fidelity and image derivatives which will work under the dual formulation.

Unlike the smoothness term which uses the L^1 norm, the data term uses an L^2 norm. The focus of the current algorithm is proposing an L^1 norm in the smoothness term. As for the data term, it was found in previous algorithms that the L^2 norm gave a good results [2]. Hence L^2 norm is used in the current data fidelity term. Starting from the non-linearised version of

the data term, an image derivatives term is inserted, hence the data term of Equation-4.24 is written as follows:

$$E_{Primal} = \int_{\Omega} \Big(\big(\alpha |I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})|^2 + \gamma |\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x})|^2\big) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \Big) \quad (4.35)$$

where γ is the weight for image gradients. This Equation is the decoupled data term taken from Equation-4.24. It consist of the intensity value difference ($|I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})|^2$), and the gradient value difference ($|\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x})|^2$). In addition to the decoupling term ($\frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2$). The aim is to minimise the intensity difference between the first image and the warped second image. The warped second image can be written in the following form:

$$I_2(\mathbf{x}+\mathbf{u}) = I_2(\mathbf{x}+\mathbf{u}+\mathbf{u_0}-\mathbf{u_0})$$

where $\mathbf{u_0}$ is the initial displacement of the pixels. The linearised version of the image intensity difference can be written in the following form:

$$I_2(\mathbf{x}+\mathbf{u}) = I_2(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_2(\mathbf{x}+\mathbf{u_0}).$$

Hence, the intensity difference term in Equation-4.35 can be written as follows:

$$I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x}) = I_{t0} + (\mathbf{u}-\mathbf{u_0})\nabla I_2(\mathbf{x}+\mathbf{u_0}) \quad (4.36)$$

where $I_{t0} = I_2(\mathbf{x}+\mathbf{u_0}) - I_1(\mathbf{x})$ is the initial intensity difference between the first image and the warped second image. The image gradient term can be written as follows:

$$\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x}) = \big(I_{2x}(\mathbf{x}+\mathbf{u}), I_{2y}(\mathbf{x}+\mathbf{u})\big)^{\mathsf{T}} \quad (4.37)$$

Similar to Equation-4.36, image gradient can be approximated to have the following form:

$$I_{2x}(\mathbf{x}+\mathbf{u}) = I_{2x}(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_{2x}(\mathbf{x}+\mathbf{u_0}). \quad (4.38)$$

$$I_{2y}(\mathbf{x}+\mathbf{u}) = I_{2y}(\mathbf{x}+\mathbf{u_0}) + (\mathbf{u}-\mathbf{u_0})\nabla I_{2y}(\mathbf{x}+\mathbf{u_0}). \quad (4.39)$$

Plugging all these terms into Equation-4.35:

$$E_{data} = \int_{\Omega} \Big(\alpha \big| I_{t0} + (\mathbf{u}-\mathbf{u_0})\nabla I_2 \big|^2 + \gamma \big| \big(I_{tx} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2x} \big), \big(I_{ty} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2y} \big) \big|^2 \Big) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \tag{4.40}$$

With the following abbreviations used:

$$\begin{aligned} I_{2x} &= \frac{\partial}{\partial x} I_2(\mathbf{x}+\mathbf{u_o}) \\ I_{2y} &= \frac{\partial}{\partial y} I_2(\mathbf{x}+\mathbf{u_o}) \\ I_{tx} &= \frac{\partial}{\partial x} I_2(\mathbf{x}+\mathbf{u_0}) - \frac{\partial}{\partial x} I_1(\mathbf{x}) \\ I_{ty} &= \frac{\partial}{\partial y} I_2(\mathbf{x}+\mathbf{u_0}) - \frac{\partial}{\partial y} I_1(\mathbf{x}) \end{aligned} \tag{4.41}$$

The solution of the primal step of Equation-4.24 requires the minimisation of $E_{Primal}(\mathbf{z})$, where:

$$E_{Primal}(\mathbf{z}) = \int_{\Omega} \Big(\alpha \big| I_{t0} + (\mathbf{z}-\mathbf{u_0})\nabla I_2 \big|^2 + \gamma \big| \big(I_{tx} + (\mathbf{z}-\mathbf{u_0})\nabla I_{2x} \big), \big(I_{ty} + (\mathbf{z}-\mathbf{u_0})\nabla I_{2y} \big) \big|^2 \Big) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \tag{4.42}$$

Equation-4.42 can be solved by setting the derivatives with respect to $zx,\ zy$ equal to zero. The derivation with respect to zx yields the following equation:

$$\begin{aligned} \big[(\alpha I_{2x}^2 + \gamma I_{2xx}^2 + \gamma I_{2xy}^2) + \frac{1}{\theta} \big] zx \\ + \big[\alpha I_{2x} I_{2y} + \gamma I_{2xy} (I_{2xx} + I_{2yy}) \big] zy \\ = -\big(\alpha r_{t0} I_{2x} + \gamma r_{tx0} I_{2xx} + \gamma r_{ty0} I_{2xy} \big) + \frac{u}{\theta}. \end{aligned} \tag{4.43}$$

Similarly a derivation with respect to zy yields the following equation:

$$\begin{aligned}\big[\big(\alpha I_{2x}I_{2y}+\gamma I_{2xx}I_{2xy}+\gamma I_{2xy}I_{2yy}\big)\big]zx\\ +\big[\big(\alpha I_{2y}^2+\gamma I_{2xy}^2+I_{2yy}^2\big)+\frac{1}{\theta}\big]zy\\ =-\big(\alpha r_{t0}I_{2y}+\gamma r_{tx0}I_{2xy}+\gamma r_{ty0}I_{2yy}\big)+\frac{v}{\theta}.\end{aligned} \tag{4.44}$$

with the following abbreviation used:

$$\begin{aligned}I_{2xx}&=\frac{\partial^2}{\partial x^2}I_2(\mathbf{x}+\mathbf{u_0})\\ I_{2yy}&=\frac{\partial^2}{\partial y^2}I_2(\mathbf{x}+\mathbf{u_0})\\ I_{2xy}&=\frac{\partial}{\partial y}\frac{\partial}{\partial x}I_2(\mathbf{x}+\mathbf{u_0})\\ r_{t0}&=I_{t0}-u_0I_{2x}-v_0I_{2y}\\ r_{tx0}&=I_{tx}-u_0I_{2xx}-v_0I_{2xy}\\ r_{ty0}&=I_{ty}-u_0I_{2xy}-v_0I_{2yy}\end{aligned} \tag{4.45}$$

4.3.3 Robust Data Term

The data term in the optical flow energy function is based on the assumption that the illumination does not change between successive images. As discussed earlier this assumption is not always valid. In order to improve the robustness of the data term in the presence of such changes, a robust penalisation function is used to mitigate the penalisation. A robust function of the form $\Psi(s^2)=\sqrt{s^2+\varepsilon^2}$ is used in the data term (see Equation-4.35), this renders the data term to have the following form:

$$\begin{aligned}E_{Primal}=\int_\Omega\Big(\alpha\,\Psi\big(|I_2(\mathbf{x}+\mathbf{u})-I_1(\mathbf{x})|^2\big)+\gamma\,\Psi\big(|\nabla I_2(\mathbf{x}+\mathbf{u})-\nabla I_1(\mathbf{x})|^2\big)+\\ \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2\Big)\end{aligned} \tag{4.46}$$

The data and the gradients terms are linearised (see Equation-4.40, Equation-4.41). Hence,

Equation-4.46 is written in the following way:

$$E_{primal} = \int_{\Omega} \alpha\, \Psi\Big(\big|I_{t0} + (\mathbf{u}-\mathbf{u_0})\nabla I_2\big|^2\Big) + \gamma\, \Psi\Big(\big|\big(I_{tx} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2x}\big), \big(I_{ty} + (\mathbf{u}-\mathbf{u_0})\nabla I_{2y}\big)\big|^2\Big)\Big) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \tag{4.47}$$

The minimisation is found by setting the derivatives of $E_{primal}(\mathbf{z})$ with respect to zx, zy equal to zero in a similar way as was done in the previous section (Section-4.3.2). Equation-4.47 is written as follows:

$$E_{primal} = \int_{\Omega} \alpha\, \Psi\Big(\big|I_{t0} + (\mathbf{z}-\mathbf{u_0})\nabla I_2\big|^2\Big) + \gamma\, \Psi\Big(\big|\big(I_{tx} + (\mathbf{z}-\mathbf{u_0})\nabla I_{2x}\big), \big(I_{ty} + (\mathbf{z}-\mathbf{u_0})\nabla I_{2y}\big)\big|^2\Big)\Big) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \tag{4.48}$$

After finding the derivatives, a set of equations are obtained which can be easily solved to find zx, zy. Analysing the derivative with respect to zx yields the following equation:

$$\begin{aligned} \big[\alpha\, \Psi'_1.I_{2x}^2 + \gamma\, \Psi'_2(I_{2xx}^2 + I_{2yx}^2) + \frac{1}{\theta}\big] zx \\ + \big[\alpha\, \Psi'_1.I_{2x}I_{2y} + \gamma\, \Psi'_2.I_{2xy}(I_{2xx} + I_{2yy})\big] zy \\ = -\big[\alpha\, \Psi'_1 r_{t0} I_{2x} + \gamma\, \Psi'_2 r_{tx0} I_{2xx} + \gamma\, \Psi'_2 r_{ty0} I_{2xy}\big] + \frac{u}{\theta}. \end{aligned} \tag{4.49}$$

Similarly derivation with respect to zy yields the following equation:

$$\begin{aligned} \big[\alpha\, \Psi'_1.I_{2x}I_{2y} + \gamma\, \Psi'_2 I_{2yx}(I_{2xx} + I_{2yx})\big] zx \\ + \big[\alpha\, \Psi'_1.I_{2y}^2 + \gamma\, \Psi'_2.(I_{2xy}^2 + I_{2yy}^2)\big] zy \\ = -\big[\alpha\, \Psi'_1 r_{t0} I_{2y} + \gamma\, \Psi'_2 r_{tx0} I_{2xy} + \gamma\, \Psi'_2 r_{ty0} I_{2yy}\big] + \frac{v}{\theta} \end{aligned} \tag{4.50}$$

where Ψ' is the derivative of Ψ, and the abbreviation in Equation-4.45 is used.

Equation-4.49 and Equation-4.50 are two simultaneous equations. The unknown terms are only zx and zy which are required to be found. The rest of the terms are already known and can be computed in a straight forward manner. The values of zx and zy can be found via any simultaneous equation solution method, such as Successive over relaxation [155] (used in Section-3.39 to compute (u,v) in the CLG optical flow) or direct division as used in the MATLAB implementation for this thesis. Here Ψ_1, Ψ_2 are defined as follows:

$$\Psi_1 = \Psi\Big(\big|I_{t0} + (\mathbf{z} - \mathbf{u_0})\nabla I_2\big|^2\Big)$$
$$\Psi_2 = \Psi\Big(\big|\big(I_{tx} + (\mathbf{z} - \mathbf{u_0})\nabla I_{2x}\big), \big(I_{ty} + (\mathbf{z} - \mathbf{u_0})\nabla I_{2y}\big)\big|^2\Big)$$

Hence, optimisation for Equation-4.24 can be preformed in primal-dual steps. The aim of the dual step is to minimise Equation-4.25, via fixed-point iteration of Equation-4.33, and Equation-4.33. To solve the primal step, Equation-4.35 is minimised via point-wise solution of Equation-4.43, and Equation-4.43.

4.3.4 Colour Realisation

The data term that has been used so far works with grey-scale images. Although grey-scale images show good results in the estimation of optical flow [14], [2], the use of colour images will improve the performance due to the richer photometric information they encompass [72], [73]. In RGB images, the colours are encoded in three values of Red, Green and Blue. The RGB is an additive colour model, where the colour of each pixel is decided based on combination values of these three colours [141]. Hence, the images can be expressed in terms of these three channels, and can be written in the following way $I1(x,y) = \big(I_1^{c1}(x,y), I_1^{c2}(x,y), I_1^{c2}(x,y)\big)$, and $I2(x,y) = \big(I_2^{c1}(x,y), I_2^{c2}(x,y), I_2^{c2}(x,y)\big)$. In this section, the optical flow estimation is extended in order to handle colour images.

Colours in this space are represented in Cartesian coordinate system. Colours at each pixel can be viewed as a combination of the three primary colours (i.e. Red, Green and Blue). Images can be converted to their intensity values (grey-scale) as a weighted sum of the three colour values. According to the recommendations of ITU-R BT.601-7[1] the following weights are used:

$$I_{intensity} = 0.299 * R + 0.587 * G + 0.114 * B$$

[1] International Telecommunication Union - Recommendations. BT.601-7 https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en

Where R, G, B are the three components (Red, Green, Blue). Images from the Middlebury dataset are available in RGB colours. In order to make use of the additional information conveyed in the three colour channels, it is possible to use the three channels to separately estimate the minimisers of (zx, zy), as explained in this section.

The data term Equation-4.46 can be extended to incorporate the three colour channels in the RGB colour model. A Robust data term is used here. This equation can be written in the following form:

$$E_{Primal} = \int_{\Omega} \Big(\alpha \, \Psi\big(|I_2^c(\mathbf{x}+\mathbf{u}) - I_1^c(\mathbf{x})|^2\big) + \gamma \, \Psi\big(|\nabla I_2^c(\mathbf{x}+\mathbf{u}) - \nabla I_1^c(\mathbf{x})|^2\big) + \frac{1}{2\theta}(\mathbf{u}^c - \mathbf{z}^c)^2 \Big) \tag{4.51}$$

where $c \in \{c_1, c_2, c_3\}$ are the three colour channels in the RGB colour model.

The minimisation for this equation is similar to the minimisation of the data term in the grey-scale images case. Hence the minimisation is found by setting the derivatives with respect to zx, zy equal to zero. The following set of equations are obtained, which can be solved easily to find the values of zx, zy.

$$\begin{aligned} &\big[\alpha \, (\Psi_1^c)'.(I_{2x}^c)^2 + \gamma \, (\Psi_2^c)'((I_{2xx}^c)^2 + (I_{2yx}^c)^2) + \frac{1}{\theta}\big] zx^c \\ &+ \big[\alpha \, (\Psi_1^c)'.I_{2x}^c I_{2y}^c + \gamma \, (\Psi_2^c)'.I_{2xy}^c (I_{2xx}^c + I_{2yy}^c)\big] zy^c \\ &= -\big[\alpha \, (\Psi_1^c)' r_{t0}^c I_{2x}^c + \gamma \, (\Psi_2^c)' r_{tx0}^c I_{2xx}^c + \gamma \, (\Psi_2^c)' r_{ty0}^c I_{2xy}^c\big] + \big(\frac{u}{\theta}\big)^c. \end{aligned} \tag{4.52}$$

$$\begin{aligned} &\big[\alpha \, (\Psi_1^c)'.I_{2x}^c I_{2y}^c + \gamma \, (\Psi_2^c)' I_{2yx}^c (I_{2xx}^c + I_{2yx}^c)\big] zx^c \\ &+ \big[\alpha \, (\Psi_1^c)'.(I_{2y}^c)^2 + \gamma \, (\Psi_2^c)'.((I_{2xy}^c)^2 + (I_{2yy}^c)^2)\big] zy^c \\ &= -\big[\alpha \, (\Psi_1^c)' r_{t0}^c I_{2y}^c + \gamma \, (\Psi_2^c)' r_{tx0}^c I_{2xy}^c + \gamma \, (\Psi_2^c)' r_{ty0}^c I_{2yy}^c\big] + \big(\frac{v}{\theta}\big)^c. \end{aligned} \tag{4.53}$$

The values of $(\mathbf{u}, \mathbf{z})$ are replicated at each iteration to cope with the three channels of the images, and thus to obtain $(\mathbf{u}^c, \mathbf{z}^c)$. Additionally the values of $(\mathbf{u}^c, \mathbf{z}^c)$ are averaged before starting the dual step (see Section-4.3.1).

4.4 Summary

In this chapter a steered-L^1 norm is introduced. This norm can improve the performance of the optical flow algorithm in terms of accuracy. The conventional total variation L^1 norm is isotropic in nature, hence it decreases penalisation at locations near motion edges. This in fact decreases the filling-in effect along motion edges. This filling-in effect is one of the properties of global optical flow algorithms, it enables the estimation of displacements at points in untextured areas. The steered-L^1 formulated in this chapter decreases penalisation across motion edges and increase it along these edges, this in effect improves the filling-in effect and increase displacement field accuracy. The steering of the smoothness term was done by multiplying with eigenvectors of the structure tensor of local image structures. Although eigenvectors of the structure tensor to improve the performance of optical flow algorithms has been used before in other smoothness terms [15], adopting this idea to the L^1 norm is non-trivial. This is due to the fact that the L^1 norm is not continuously differentiable. Additionally, this chapter provided the formulation for a robust data term with delayed linearisation, and the addition of an image gradient term to increase robustness against illumination changes in a similar manner to the algorithm in [2] . This data term is incorporated in primal-dual formulation.

Experiments on the proposed algorithm in this chapter are conducted in Chapter-5. The experiments are designed to investigate each part of the proposed algorithm. Results of the experiments are reported qualitatively using colour-coded results to enable visual inspection of the estimated displacement field. Additionally results are reported quantitatively where ground truth data are available.

Chapter 5

Experiments and Results

In the previous chapters several optical flow algorithms were investigated, and an improved algorithm was proposed. In particular, a steered-L^1 norm smoothness term was proposed to improve the efficiency of the optical flow algorithm. The proposed steered-L^1 norm directionally penalises the flow field depending on the orientation of image structures, and with magnitude proportional to the magnitude of the flow field itself. This can also be referred to as joint image-flow driven smoothness [15], which improves the edge conservation properties of the algorithm compared with the second order norm. In addition to that the proposed enhanced smoothness increases the filling-in effect observed in global optical flow algorithms, which leads to improved flow field estimation accuracy. Although similar smoothness terms have been used, the contribution here is using it with the robust L^1 norm.

In general the efficiency of optical flow algorithms are assessed using benchmark datasets, and compared against each other using error metrics. Several datasets were used over previous years such as, the Middlebury dataset benchmark [6], and MPI-Sintel [150].

In this chapter several experiments are conducted to study the performance of the smoothness term on the calculated flow field. Experiments are carried out using the Middlebury dataset [6], and compared to other algorithms qualitatively using the colour code discussed previously, and quantitatively via the error metrics benchmarks. This chapter is divided into two sections. The first section is dedicated to investigate the image registration based on the CLG optical flow method. Experiments conducted using this method demonstrate an improved image registration method when compared with the global optical flow method [1]. In the second section, several experiments are used to demonstrate the performance of the proposed optical flow method with the steered-L^1 norm. Each experiment aims to highlight the contribution that each constituent has made to the final method.

5.1 CLG Image registration

In Chapter-3 an improved CLG optical flow algorithm was proposed, where a bi-lateral filter is used to calculate the structure tensor resulting in an improved edge-preserving and accuracy of the calculated flow field. In this section experiments are conducted to demonstrate the performance of image registration using the CLG method. The experiments here can be considered as an extension and further investigation of the initial results demonstrated in Section-3.7. Image registration using bi-lateral CLG optical flow is compared with the registration of the global optical flow proposed by Horn-Schunck [1]. The Middlebury benchmark [6] is used in these experiments.

5.1.1 Implementation

Algorithm-3 illustrates the implementation.

Algorithm 3: Implementation Algorithm of bi-lateral CLG optical flow.

Input: Images I_1& I_2,
number of pyramid layers L, current layer l,
flow at each warp (du, dv)
Create pyramid of images with 5 layer, and a ratio of 0.5;
initialization;
l=1; initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;
while $l \leqslant L$ **do**
  Up-scale size of (u_{l-1}, v_{l-1}) to (u_l, v_l);
  while *No. of Warps* $\leqslant$ *4* **do**
    Warp I_{2l}, I_{2x}, I_{2y} towards I_{1l}, I_{1x}, I_{1y};
    while *No. of fixed_point_iterations* <= *5* **do**
      Update $\Psi'_1,\ \Psi'_2$;
      Calculate du, dv ;
      Update the flow $u_l = u_l + du,\ v_l = v_l + dv$;
    end
  end
  $u = u_l,\ v = v_l$;
end
Output: (Displacement field (u, v))

The aforementioned algorithm is implemented in MATLAB. Image sequences are resized several times to create the image pyramid, the minimisation is performed in a C2F framework. Each layer of the pyramid is resized with a ratio of 0.5 of the previous image layer. Images and displacement fields rescaling are done using bi-cubic interpolation. A structure tensor is computed at every layer using a bi-lateral filter.

To compare the performance and the robustness of both optical flow algorithms, zero-mean Gaussian noise is added to the image sequences with different variances. The displacement fields are obtained using the aforementioned algorithms, then these displacement fields are used to register the second image to the first image of the sequence. This is done by warping the second image using the estimated displacement fields. For the warping process it is important to point out that the images used are the original images (i.e without added noise), this is because it is easier to visually investigate the registered image to be compared with the reference image.

5.1.2 Results

Estimating Optical Flow

The first set of experiments are conducted to compare the performance of the two methods (i.e. Horn-Schunck and the bi-lateral CLG optical flow). The following table illustrates the AAE results for both methods of evaluation using Middlebury dataset [6].

Image sequence	Horn-Schunck	Bi-lateral CLG
RubberWhale	8.10	6.99
Dimetrodon	3.90	4.14
Urban2	6.20	8.46
Urban3	14.50	14.31
Venus	9.36	8.13
Grove2	4.02	2.94
Grove3	8.10	7.89
Hydrangea	7.72	7.50
Average	7.74	7.55

Table 5.1 AAE ^{o} comparison between Horn-Schunck and the implemented bi-lateral CLG

This table depicts the results obtained by applying the two methods to all the eight training

sequences of the Middlebury dataset. The AAE is calculated as discussed earlier (Equation-3.23) between the computed flow field and the ground truth. The smaller AAE value indicates more accurate flow field computation, as these values mean that the estimated flow field is closer to the ground truth. It can be seen in this table that the proposed bi-lateral CLG yield better results in comparison with the Horn-Schunck method.

Image Registration Performance

In this section the performance of the image registration of the bi-lateral CLG is assessed, and compared with image registration of the global method of Horn-Schunck [1]. The CLG optical flow offers a more accurate and more robust method to estimate optical flow displacement fields in the presence of noise. It is desired to test the robustness of the image registration method. To this end zero-mean Gaussian noise is added to the image sequences. The registration quality is then visually assessed using the results obtained by applying the two methods.

To perform the registration, the dense optical flow field is calculated via the two methods. The displacements field is then used to register the uncontaminated second image in the sequence via bi-cubic warping. In Figure-5.1 a first image of an image sequence is shown, this image is contaminated with a Gaussian noise with a variance of 0.005.

Fig. 5.1 DogDance image sequence with added noise.
Left: Original image. Right: Image with zero-mean Gaussian noise with variance of 0.005 added.

The optical flow displacement field is calculated for the contaminated image sequence. The displacement field is estimated using both optical flow methods (i.e. Horn-Schunck and

bi-lateral CLG). Then these displacements are used to register the second image (of the sequence) to the first image. Figure-5.2 depicts the results of the registration.

Fig. 5.2 Image registration comparison of the two optical flow methods.
Images are registered using the estimated optical flow filed. Left: Horn-Schunck. Right: Bi-lateral CLG.

To further investigate the difference, Figure-5.3 illustrates zoomed-in parts of the registered images.

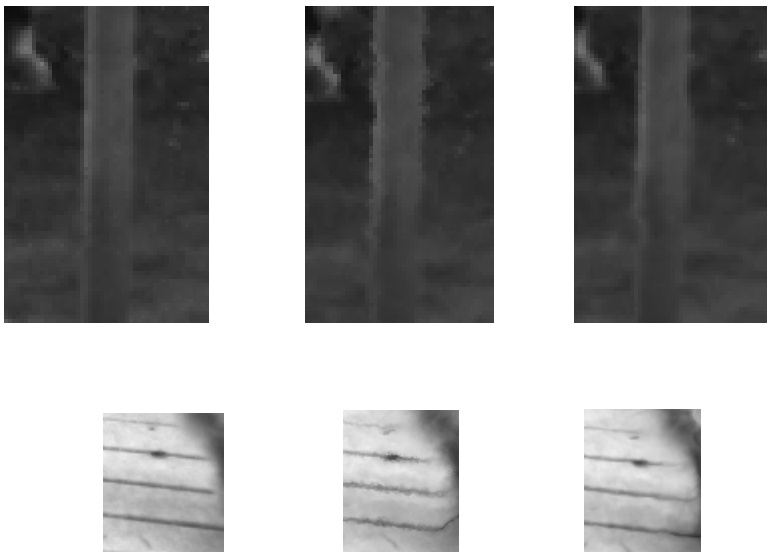


Fig. 5.3 Image registration comparison with zoomed-in image segments.
Images are registered using the estimated optical flow field.
Left: Original image. Middle: Horn-Schunck. Right: Bi-lateral CLG.
It can be noticed that images registered via bi-lateral CLG optical flow are more robust as this method is able to preserve lines and edges.

By examining the previous image, the differences between the images can be noticed. While the image registered using the Horn-Schunck displacement field can be seen with blurry edges, the image registered with the bi-lateral CLG is able to better preserve image edges.

Figure-5.4 shows a depiction of the results of registration using several images. The registration was implemented using displacement fields estimated via the two optical flow methods, however, this time with a zero-mean Gaussian noise of variance of 0.05.

Fig. 5.4 Image registration comparison of several image sequences.
These images are contaminated with zero-mean Gaussian noise of 0.05 variance.
Left column: Original image. Middle column: Horn-Schunck. Right column: Bi-lateral CLG.

Image registration has a wide application in image processing and computer vision, an ac-

curate and robust registration is very important. Although the experiments conducted in this chapter use images of general nature, further work can be carried out to include images of certain type of registration. In addition to the use of the CLG optical flow, the CLG optical flow is modified by using a bi-lateral filter to enhance the edge preserving performance of the algorithm. An example of such applications is medical images, where image registration is used to obtain deformable dense results. Noise can have a measurable effect on the results of registration. In medical imaging, noise also can have a tangible effect on the results of registration (for example Gaussian noise [160]). In this section experiments were carried to illustrate the difference in the quality of registration compared to the Horn-Schunk optical flow method. It is going to be interesting to apply this method on different types of images such as medical images and to understand how it can improve the registration quality in terms of noise robustness and edge preservation.

The size of the filter in the data fidelity term is important factor that affects the performance of the algorithm. In the CLG algorithm that was proposed Bruhn et al. [12], the authors used Gaussian filter in the data term. The size of the filter has an impact on the performance. A small filter size (with small standard deviation) may not have the desired effect in terms of illumination change robustness. On the other hand a large filter may include regions from across motion boundaries, which may lead to over-smoothing of these boundaries. If a bi-lateral filter is used, such as the one in the current experiments, this over-smoothing is mitigated due to assigning lower weights for different colour pixels across the boundaries. However these will still have effect on the computation, especially if the colours across the boundaries are similar. In the current experiments the bi-lateral filter chosen has a 5×5 kernel size with standard deviation of 2, and a standard deviation for the intensity similarity of 5/3. Several parameters were tested, it was found that these parameters gave good results, hence it was chosen for the experiments.

5.2 Steered-L^1 algorithm

In this section several experiments are conducted[1], and the purpose is to demonstrate the performance of the proposed steered-L^1 optical flow algorithm. In the first part of this section, the implementation of this algorithm in MATLAB is discussed. This is followed by experiments to demonstrate the effect of using the steered-L^1 norm on the overall performance of the algorithm (Section-4.3.1). Then experiments are conducted to demonstrate the effect of using the robust data term in two cases, grey-scale and colour images (see Section-

[1] The results of the steered-L^1 are published in a recent paper at the WSCG 2016

4.3.3, Section-4.3.4). In the following subsection a new intermediate filtering is presented with the aim to improve the accuracy of optical flow estimation. Experiments show that the performance of the optical flow algorithm proposed in this thesis is improved using the extended intermediate filtering.

5.2.1 Implementation

In this section, the implementation of the algorithm with its variants is discussed. In the first part, the implementation the total variation L^1 optical flow is explained. In this implementation the data term used is the one proposed in Section-4.3.2, and the L^1 smoothness term [22]. In the second part the implementation of the steered-L^1 is discussed.

L^1 implementation

The implementation of the algorithm with its variants was done using MATLAB. Since the algorithm is variational, only small displacements can be computed, hence minimisation was performed under the C2F framework. A fine resolution image pyramid is used here with 80 layers, and down-sampling ratio of each layer is 0.95 of the previous resolution. The down-sampling is performed using bi-cubic interpolation, and the same interpolation method is used for the displacement field up-sampling. The x and y first and second derivatives are approximated via the filter kernel $[-1,\ 9,\ -45,\ 0,\ 45,\ -9,\ 1]/(60)$ [12]. The image derivatives are computed at each layer of the pyramid, the derivative used is the average of the derivatives of the first image and the warped second image, this step improves the results [143]. The divergence and derivative for the variable $\mathbf{p}$ are approximated using the three point kernel $[-1,\ 0,\ 1]$.

For each pyramid layer six warps are applied, for each warp a median filter is used to remove outliers in the computed flow field. Algorithm-4 illustrates the implementation.

Algorithm 4: Implementation of L^1 norm with data term proposed in Section-4.3.2.

Input: Images I_1& I_2, number of pyramid layers $L = 80$, current layer l , number of warps w=6

number of iterations per warp It = 20

Create pyramid of images with L layer, and a ratio of 0.95;

initialization;

l=1;

initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;

while $l \leqslant L$ **do**

  Up-scale size of (u_{l-1}, v_{l-1}) to (u_0, v_0);

  while *No. of warps* $\leqslant w$ **do**

    Warp I_{2l}, I_{2x}, I_{2y} towards I_{1l}, I_{1x}, I_{1y};

    while *No. of iterations* $\leqslant It$ **do**

      Calculate the auxiliary variable zx, zy;

      Use these values to calculate the flow u_l, v_l;

      Update p_u, p_v;

    end

    Use median filter to remove outliers from u_l, v_l;

  end

end

Output: (Displacement field (u, v))

Steered-L^1

The implementation of the steered-L^1 is similar to the previous implementation with some differences. At each pyramid layer the structure tensor (see Equation-3.20) is computed. In order to compute the derivatives to obtain Equation-3.20 a 5×5 optimised derivative filter D [40] is used:

$$D = (0.0234, 0.2415, 0.4700, 0.2415, 0.0234)^{\mathsf{T}} * (0.0838, 0.3323, 0, -0.3323, -0.0838) \tag{5.1}$$

The Gaussian kernel used for the structure tensor computation is a 6×6 with standard deviation of 2. After calculating the structure tensor, eigen-decomposition is performed to find the two eigenvectors. To obtain $\mathbf{p}_{su},\ \mathbf{p}_{sv}$ the following filter kernels [161] are used to compute $\mathbf{p}_u,\ \mathbf{p}_v$:

$$h_x = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix} \tag{5.2}$$

$$h_y = \frac{1}{32}\begin{bmatrix} -3 & 10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} \tag{5.3}$$

where h_x, h_y are the kernels to estimate the x, y derivatives respectively. The implementation of the steered-L^1 is illustrated in Algorithm-5.

Algorithm 5: Implementation Algorithm of steered-L^1 norm.

Input: Images I_1& I_2, number of pyramid layers $L = 80$, current layer l , number of warps w=6 , number of iterations per warp It = 20

Create pyramid of images with L layer, and a ratio of 0.95;
initialization;
l=1;
initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;
while $l \leqslant L$ **do**
    Up-scale size of (u_{l-1}, v_{l-1}) to (u_0, v_0);
    Find structure tensor of the first image I_1;
    while *No. of warps* $\leqslant w$ **do**
        Warp I_{2l}, I_{2x}, I_{2y} towards I_{1l}, I_{1x}, I_{1y};
        while *No. of iterations* $\leqslant It$ **do**
            Calculate the auxiliary variable zx, zy;
            Use these values to calculate the flow u_l, v_l;
            Calculate p_{su}, p_{sv}, use them to find p_u, p_v;
        end
        Use median filter to remove outliers from u_l, v_l;
    end
end
Output: (Displacement field (u, v))

5.2.2 Parameters Variations

It is important to carefully select the parameters and weights of different terms in any optical flow algorithms. Parameters affect the performance of the algorithms in terms of accuracy of flow field estimation. Therefore it is crucial to find the parameters values which gives the best results prior to assessing the work of any algorithm. To this end several experiments were conducted to find the optimum parameters for the algorithm. The focus in this section is to study the effect of varying several parameters on the performance of the current algorithm. Namely α which is the data term weight parameter (see Equation-2.1), and γ which is the weight of the image gradients in the data term (see Equation-4.35). The data term used here is the one derived in this thesis (see Section-4.3.2). The Average End-Point Error (AEPE) is used here rather than the Average Angular Error (AAE). As discussed in Section-3.4, AAE does not penalise error in a uniform way, where large errors are penalised less severely.

In the first set of experiments the value of α is varied and the resulting AEPE is calculated. Three image sequences are used here taken from the Middlebury dataset [6], and depicted here in Figure-5.5.

Fig. 5.5 First frames of three image sequences.
These are frame-10 of the three images sequences. Experiments are conducted using these sequences to illustrate the effect of parameter varying on the optical flow accuracy results.

Three different image sequences with known ground truth are used. Table-5.2 illustrates the results obtained from these experiments.

α	γ	RubberWhale	Urban2	Grove2
1	1	0.12	0.52	0.19
1/10	1	0.10	0.50	0.19
1/100	1	0.10	0.50	0.20
1/500	1	0.10	0.49	0.20
1/1000	1	0.10	0.50	0.20
1/2000	1	0.10	0.49	0.20
1/3000	1	0.10	0.49	0.20
1/4000	1	0.10	0.49	0.20
1/4700	1	0.10	0.49	0.20
1/5000	1	0.10	0.50	0.20

Table 5.2 AEPE differences with varied α.
The value of γ was fixed to examine the effect of varying α.

It can be seen by examining the previous table that a changes in α result in a small or negligible change in AEPE. This indicates that the algrrithm is robust to changes in α. Two decimal place numbers are used here. Now for the same parameters the value of γ is varied, the results are illustrated in Table-5.3 next. For the sake of illustration the weight α is fixed to a value of $\frac{1}{4000}$.

α	γ	RubberWhale	Urban2	Grove2
1/4000	10	0.13	0.56	0.24
1/4000	5	0.12	0.54	0.23
1/4000	2	0.11	0.51	0.21
1/4000	1	0.10	0.49	0.20
1/4000	1/10	0.10	0.73	0.18
1/4000	1/100	0.25	1.61	0.21
1/4000	1/500	0.46	7.15	0.27

Table 5.3 AEPE differences with varied γ.
The value of α is fixed at $\frac{1}{4000}$ to examine the effect of varying γ.

A good results can be obtained at value of $\gamma = 1$, i.e. the weight of γ is equal to the smoothness weight, while increasing or decreasing the value reduces the accuracy of the results. Small values of α seems to give a good results.

As discussed earlier, tuning the parameters (weights) is very important to ensure that the best

performance of the algorithm is achieved. These parameters control the rate of contribution of each element of the equation on the results. For example higher weight of the intensity values term and lower weight for the gradient term results in trapping in local minima as it is possible to have a lot of intensity values similarities. On the other hand, lowering both weights for the intensity and gradients terms, results in a smoother flow field, including smoothness across motion boundaries, as lowering the data term weight (intensity values term + gradient term) gives more effect for the smoothness term.

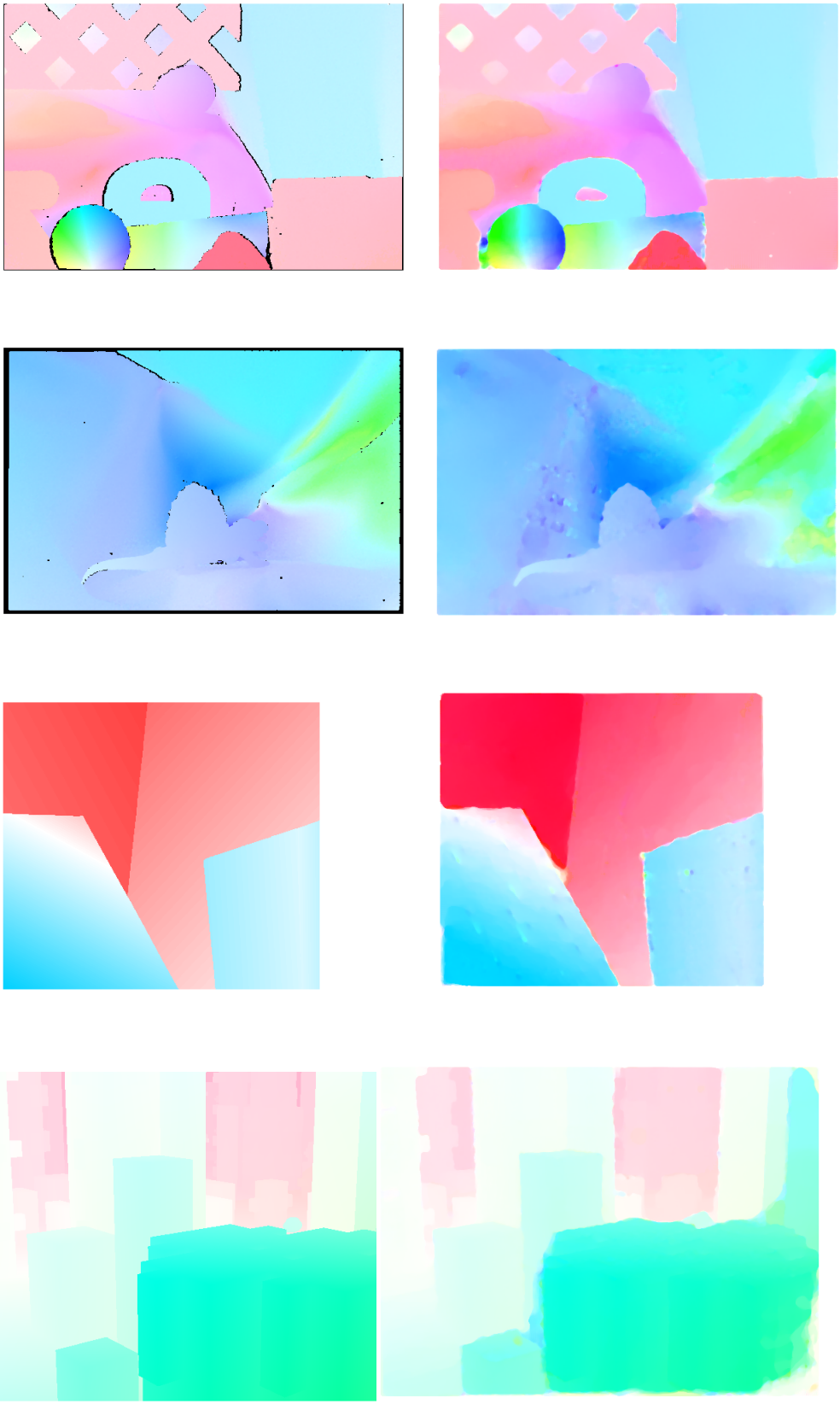
5.2.3 Steered-L^1 Norm

The experiments conducted in the previous section used the proposed data term that includes image gradient to improve robustness against illumination changes, and the smoothness term used is the total variation L^1 norm as was proposed in [22], [14]. Tests in the current section are designed to investigate the performance of the steered-L^1 norm, which is the algorithm proposed in Section-4.3. Experiments are conducted using the Middlebury dataset [6], and the parameters are set to the following values $\big(\alpha = \frac{1}{4700},\ \gamma = 1,\ \theta = 0.1,\ \tau = 0.1\big)$. The value of AEPE is obtained and compared in both cases, the total variation L^1 and the proposed steered-L^1 smoothness. In the proposed smoothness the direction of penalisation is steered by local image structures, while the magnitude is adapted with the flow magnitude. This results in an improved filing-in effect, and in turn decreases the average end-point error (AEPE). Additionally, the experiments from this point onward uses the robust data term discussed in Section-4.3.3. Table-5.4 next demonstrates the results of AEPE obtained by running both algorithms on the Middlebury dataset. In total there are eight training image sequences in the Middlebury benchmark, the results in the following table include the results obtained using all these training sequences.

Image	L^1	steered-L^1
RubberWhale	0.10	0.08
Dimetrodon	0.19	0.18
Urban2	0.66	0.63
Urban3	0.59	0.55
Venus	0.30	0.30
Grove2	0.19	0.17
Grove3	0.59	0.58
Hydrangea	0.18	0.16
Average	0.35	0.33

Table 5.4 AEPE ^{o} results of isotropic and anisotropic L^1 smoothness.

By examining the previous table, it can be seen that the use of the proposed smoothness term improved the optical flow estimation. It is worth mentioning here that a small reduction in AEPE results in a big difference in the ranking of Middlebury benchmarks, this is due to having a small evaluation dataset [90]. Figure-5.7 next demonstrates the colour-coded results obtained for the above experiment. The left column of the figure lists the ground truth for the eight sequences, while the right column includes the colour-coded results obtained via the proposed steered-L^1 norm.



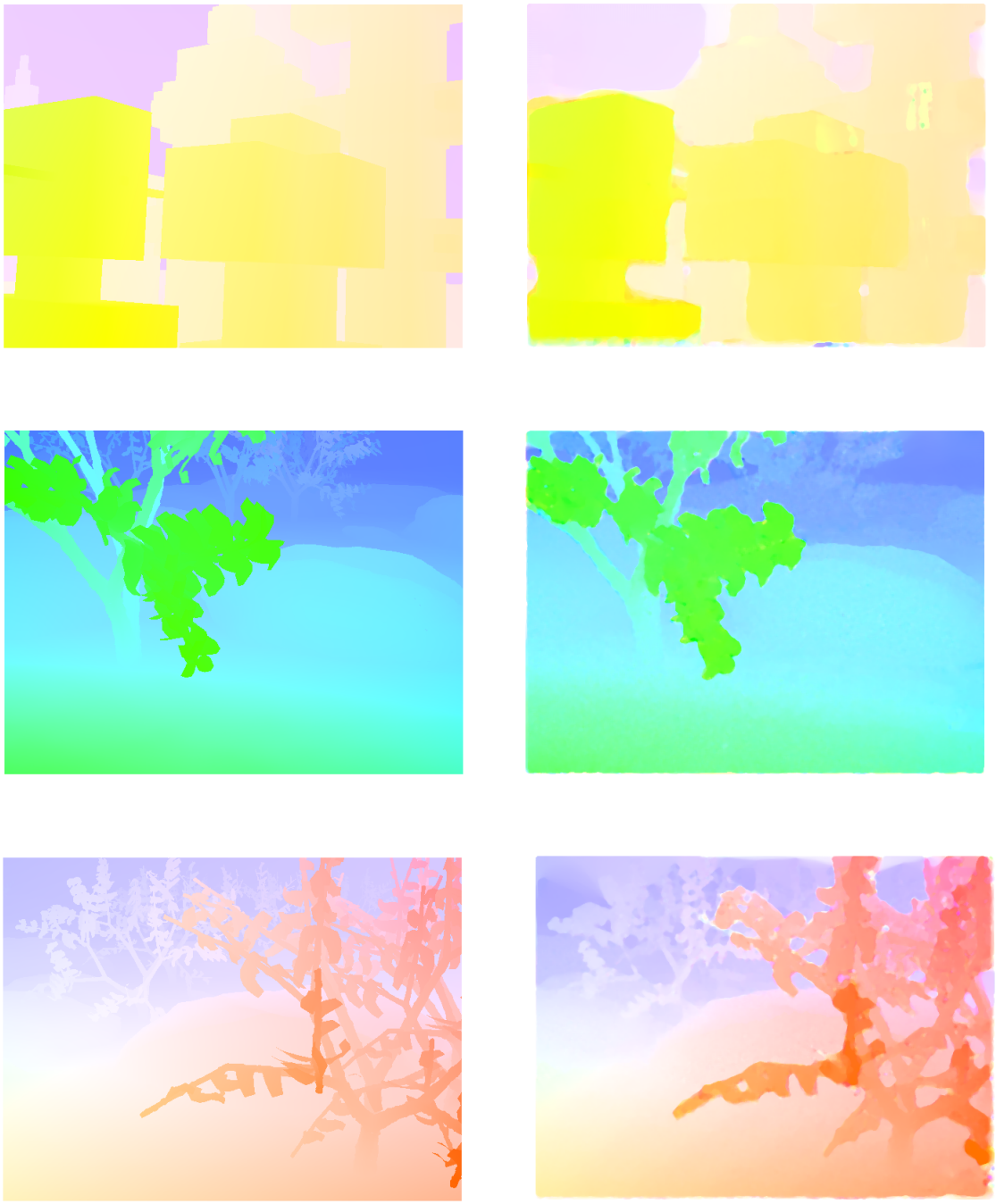


Fig. 5.7 Colour-coded results comparison between the ground truth displacement fields and the proposed algorithm using steered-L^1.
Left: Ground-truth. Right: Estimated flow field using the current algorithm.

5.2.4 Colour Images Implementation

The previous sections included the investigation the effect of the steered-L^1 norm, and the improvement it made to the performance of the optical flow, all this was done using grey-scale images. From this section onward, colour images are used. This section includes a brief discussion on the implementation of the algorithm using colour images.

As was illustrated in Section-4.3.4 it is possible to use colour images instead of the grey-scale ones. In that section the data term for the the algorithm was derived for the colour images case. The data term is duplicated for the three channels of the RGB colour space. The values of zx, zy are found in the three channels and averaged to yield a single value for these variables. Algorithm-6 depicts the computation of optical flow using colour images.

Algorithm 6: Implementation Algorithm of steered-L^1 norm using colour images.

Input: Images I_1& I_2, number of pyramid layers $L = 80$, current layer l , number of warps w=6 , number of iterations per warp It = 20

Create pyramid of images with L layer, and a ratio of 0.95;
initialization;
l=1;
initialise (u_l, v_l) to $(\mathbf{0}, \mathbf{0})$;
while $l \leqslant L$ **do**
    Up-scale size of (u_{l-1}, v_{l-1}) to (u_l, v_l);
    Find structure tensor of the first image I_1;
    while *No. of warps* $\leqslant w$ **do**
        Warp I_{2l}, I_{2x}, I_{2y} towards I_{1l}, I_{1x}, I_{1y};
        while *No. of iterations* $\leqslant It$ **do**
            calculate p_u, p_v, use them to find p_{su}, p_{sv};
            replicate the terms and update the auxiliary variable zx^c, zy^c;
            Average zx^c, zy^c to yield zx, zy;
            Update the flow u_l, v_l;
        end
        Use median filter to remove outliers from u_l, v_l;
    end
end
Output: (Displacement field (u, v))

5.2.5 Extended Intermediate Filtering

Median filtering is used in optical flow algorithms to improve the accuracy of flow field estimation. It is especially used in algorithms following dual minimisation techniques, also known as splitting algorithms, where the problem is decomposed into several sub-problems each solved separately. An example of this is the dual optimisation used in this thesis. While median filtering does not have a high impact on the results of algorithms relying on Euler-Lagrange equations, the impact is clear in duality algorithms [24]. Median filtering can be applied at each warping step to remove outliers from the estimated displacements field, thus improving the accuracy of the calculated flow field. Median filters work by replacing the displacement value with the median value in a small neighbourhood (see Section-3.2.1). One can conclude that a median filter helps to improve the accuracy of the optical flow by encouraging smoother solutions (without outliers) in the computed flow field. Additionally, one of the disadvantages of the L^1 smoothness term is that it leads to a piecewise constant solution in low textured areas [23], this results in what is known as the *'stair-casing effect'*, which can be seen as artificial boundaries in the computed flow field [96]. Obviously this can decrease the accuracy of the flow field.

Taking all the previous points into consideration, an additional smoothing step is proposed to enhance the accuracy of the optical flow computation. It is suggested here that the additional step be performed immediately after using the median filter to remove outliers. Experiments conducted using this additional filtering step are found to give improved results. The intermediate filtering is called here *'Extended Intermediate Filtering (EIF)'*, and is performed at each warping step.

One option for the extended intermediate filtering is the Gaussian filter. However, Gaussian filtering may introduce blurring of edges in the computed flow field. Another possible filter is the bi-lateral filter [59], [60], which has a better edge preserving performance (see Section-3.2.2).

Although bi-lateral filters may themselves introduce some regions with a stair-casing effect, the use of such filters in the intermediate filtering step mitigates the stair-casing effect in general. This is done (as pointed out earlier) by encouraging a smoother flow field in a small neighbourhood. Algorithm-7 is a simplified algorithm for optical flow estimation

with an illustration of EIF.

Algorithm 7: Extended Intermediate Filtering (EIF). EIF stage is highlighted in bold font.

Input: Images I_1, I_2
while $l <= L$ **do**
  while *No. of Warps* <= *6* **do**
    while *No. of iterations* <= *20* **do**
      Update the auxiliary variable zx, zy;
      Update the flow u_l, v_l;
    end
    Use median filter to update u_l, v_l;
    Use bi-later filter to update u_l, v_l;
  end
end
Output: (Displacement field (u, v))

Table-5.5 depicts a comparison of the estimated flow field with/without using the proposed extended intermediate filtering on colour images.

Image	without EIF	with EIF
RubberWhale	0.08	0.08
Dimetrodon	0.15	0.14
Urban2	0.64	0.53
Urban3	0.54	0.46
Venus	0.34	0.31
Grove2	0.17	0.17
Grove3	0.57	0.57
Hydrangea	0.16	0.16
Average	0.33	0.30

Table 5.5 AEPE results with/without extended intermediate filtering.
It is worth noting that a small change in AEPE can lead to a big difference in ranking [90].

The table above shows an improvement in algorithm performance. This can make a big difference in the ranking in benchmarks [90]. The EIF is one constituent of the proposed algorithm. The effect of this filtering step may seem small; however each constituent of the algorithm introduces part of the improvements. When all the constituent work together,

the final algorithm surpasses the performance of other algorithms sharing similar principals, this is shown in more details in Section-5.2.6.

Figure-5.8 demonstrates colour-coded optical flow displacement fields. In this figure a comparison is made between the results of optical flow estimation with/without EIF.

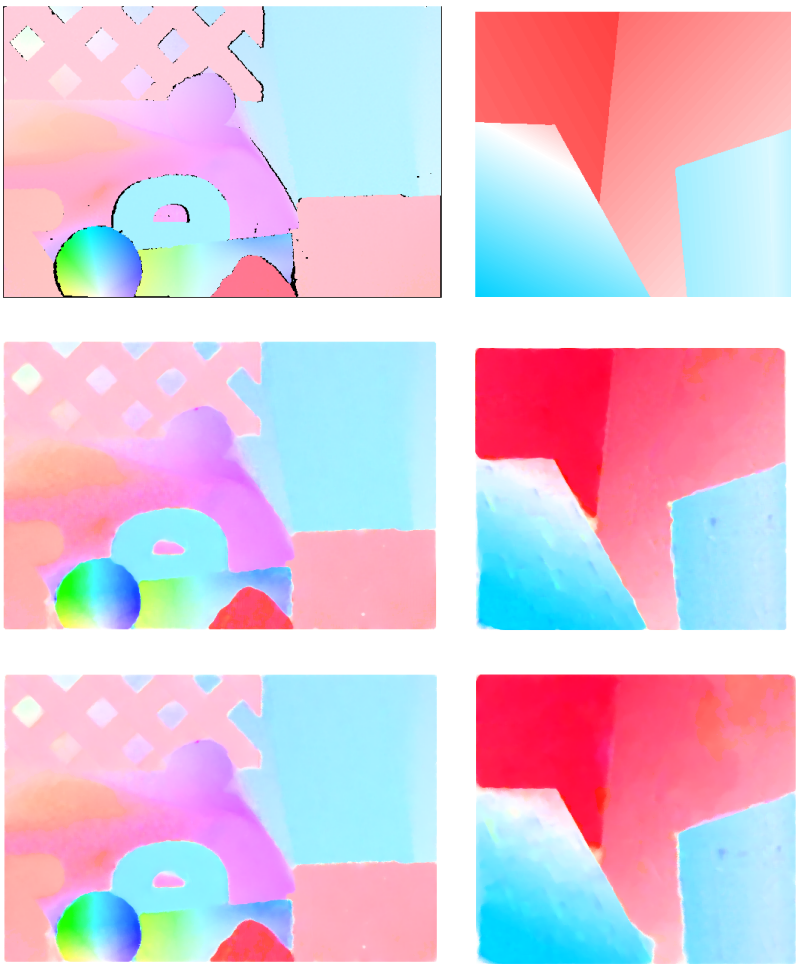


Fig. 5.8 Optical flow estimation comparison with/without EIF
Upper row: Ground truth for RubberWhale and Venus sequences. Middle row: Estimated flow field without EIF. Lower row: Estimated flow field with EIF.

To further examine the effect of EIF, Figure-5.9 demonstrates segments of optical flow.

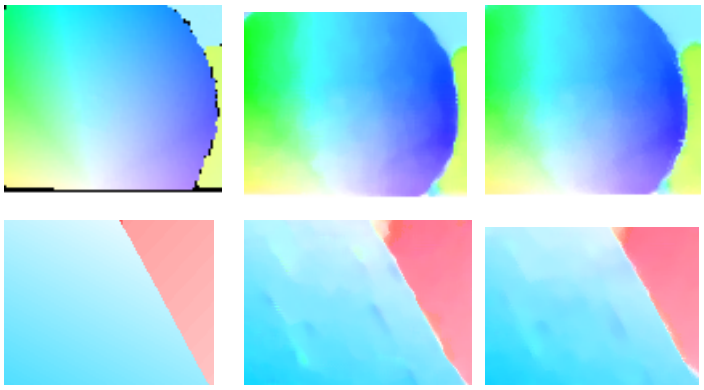


Fig. 5.9 Optical flow estimation comparison with/without EIF for selected segments
Left column: Ground truth. Middle column: Optical flow without EIF. Right column: optical flow with EIF.

The improvement that EIF introduces to the estimation of the displacement fields can be noticed by examining Figure-5.9. The optical flow field obtained using EIF is smoother and appears more similar to the ground truth than that obtained without EIF. Figure-5.10 depicts te final visualisation for the displacement fields of the Middlebury training datasets.

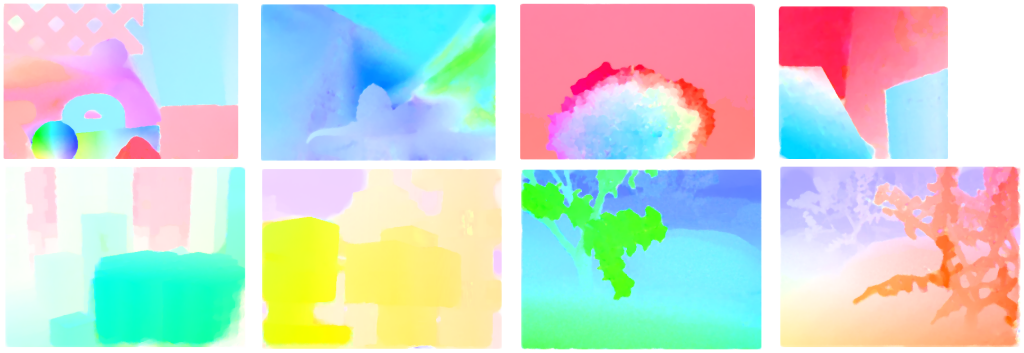


Fig. 5.10 Final optical flow field visualisation.
First row starting from left: RubberWhale, Dimetrodon, Hydrangea, Venus.
Second row starting from left: Urban2, Urban3, Grove2, Grove3

5.2.6 Comparison with Other Algorithms

In this section a comparison between the proposed algorithm and some already existing algorithms is presented. The comparison here is quantitative and uses either results reported by the authors of these algorithms, or results obtained using codes provided by the authors.

The algorithms chosen for the comparison were selected because they share similar principals to the algorithm proposed in this thesis.

The first algorithm to compare with is the LDOF proposed by Brox et al. [13]. This algorithm used delayed linearisation and augmented the data term with image gradients to increase robustness to illumination [2]. In addition to that, a descriptors' matching term was incorporated in the optical flow energy function to enable flow field detection of small objects with large displacements. The code of this algorithm was downloaded from the website provided by the authors[2]. This code is used to estimate the optical flow field of the eight training sets of Middlebury dataset.

Braux-Zin et al. [98] presented a large displacement optical flow which was inspired by the work of Brox et al. [13]. In this paper they incorporated descriptors' matching into a variational algorithm. However, the difference from [13] is the use of a primal-dual algorithm similar to the one presented in this thesis. The authors of [98] used AD-Census in the data fidelity term [99], which is one of the major differences from the algorithm presented here.

The algorithm proposed by Wedal et al. [14] shares some common features with the current algorithm. It also follows primal-dual minimisation. It was found that this algorithm has very good real-time performance. However, as pointed out in previous discussions, two major differences distinguish this algorithm from the one proposed here. First the algorithm of Wedal et al. [14] uses structure-texture decomposition to improve robustness against illumination. Unlike the algorithm proposed in this thesis which uses image gradients for this purpose, in addition to delayed linearisation. Another important difference is the use of steered-L^1 norm in the smoothness term, different from the L^1 norm used in that algorithm. The results of AEPE of this algorithm on the Middlebury training images are reported in their paper.

In Table-5.6 a comparison of the AEPE of the algorithm proposed in this thesis with the aforementioned algorithm is presented. The table includes the algorithm proposed in [98] without descriptors' matching. The results of their algorithm were reported in their paper. The results of the AEPE is shown here rounded to the nearest two decimal places.

[2] http://lmb.informatik.uni-freiburg.de/resources/software.php

Image	Braux-Zin et al. [98]	LDOF [13]	Improved $TV-L^1$ [14]	Proposed algorithm
RubberWhale	0.13	0.12	0.09	0.08
Dimetrodon	0.12	0.12	0.19	0.14
Urban2	0.46	0.33	0.32	0.53
Urban3	0.60	0.60	0.63	0.49
Venus	0.26	0.43	0.26	0.31
Grove2	0.18	0.16	0.15	0.17
Grove3	0.71	0.66	0.67	0.56
Hydrangea	0.18	0.18	0.15	0.16
Average	0.33	0.32	0.31	0.30

Table 5.6 AEPE Comparison between the proposed method and several methods. The algorithm proposed in this thesis shows an improvement by reducing AEPE for the Middlebury training image sequences.

As seen in Table-5.6, the AEPE average of the eight sequences using the proposed algorithm has improved.

The AEPE value for each frame can be interpreted as the difference in displacement between the computed optical flow and the ground truth optical flow (Equation-3.24). The Middlebury dataset is divided into training and test datasets, both include small number of image sequences (each includes 8). Therefore the results on the benchmark tends to be over-fitted and a small change in AEPE can make big leap in the ranking [89]. Although the results in Table-5.6 was obtained using the training image sequences [3], the results correlated to the test image results. For example the difference in average between LDOF [13] and the Improved $TV-L^1$ [14] is only 0.01 (Table-5.6), yet the difference is 20 positions (according to AEPE) in the Middlebury ranking table [4]

5.2.7 Results on MPI-Sintel Training Dataset

All the previous experiments were conducted using the Middlebury dataset [6]. More recently, the MPI-Sintel [150] is also being used to test the performance of optical flow algorithms (see Section-3.4.1). This dataset is more challenging due to the inclusion of large

[3] A total of 8 training sequences only available.
[4] http://vision.middlebury.edu/flow/eval/results/results-e1.php

motion and occlusion. It was reported in [150] that methods with high ranking on the Middlebury dataset have more difficulty estimating optical flow on this dataset.

The experiments conducted in the previous section were all using the training images of the Middlebury dataset. Similar experiments are to be conducted using different datasets keeping all parameters unchanged. The MPI-Sintel offers a large number of training and test frames (1064 frames for training and 564 for test). The experiments conducted here are performed using the training dataset due to the availability of ground truth data.

Results on selected frames

As stated earlier the MPI-Sintel dataset includes a large number of training frames (1064 training frames). Frames are grouped together, each group is taken from a certain clip in the film (see Section-3.4.1). In this section a demonstration of the performance of the steered-L^1 algorithm on several random frames of this dataset is presented. The following tests are conducted using a frame with a diverse image environment, including some indoor, outdoor and different lighting settings. Table-5.7 next illustrates the results obtained using the first two frames from several selected clips.

Image clip	clean	final
alley_1	0.18	0.19
ambush_5	1.72	2.78
bamboo_1	0.23	0.23
cave_2	2.01	2.29
mountain_1	0.59	1.10
shaman_2	0.11	0.13
sleeping_1	0.10	0.11
temple_1	1.15	1.93
Average	0.76	1.10

Table 5.7 AEPE for selected frames from MPI-Sintel dataset.

Figure-5.11 includes the visualised results of the MPI-Sintel dataset. The results include optical flow of two passes, the ‘clean’ and ‘final’ passes (see Section-3.4.1).

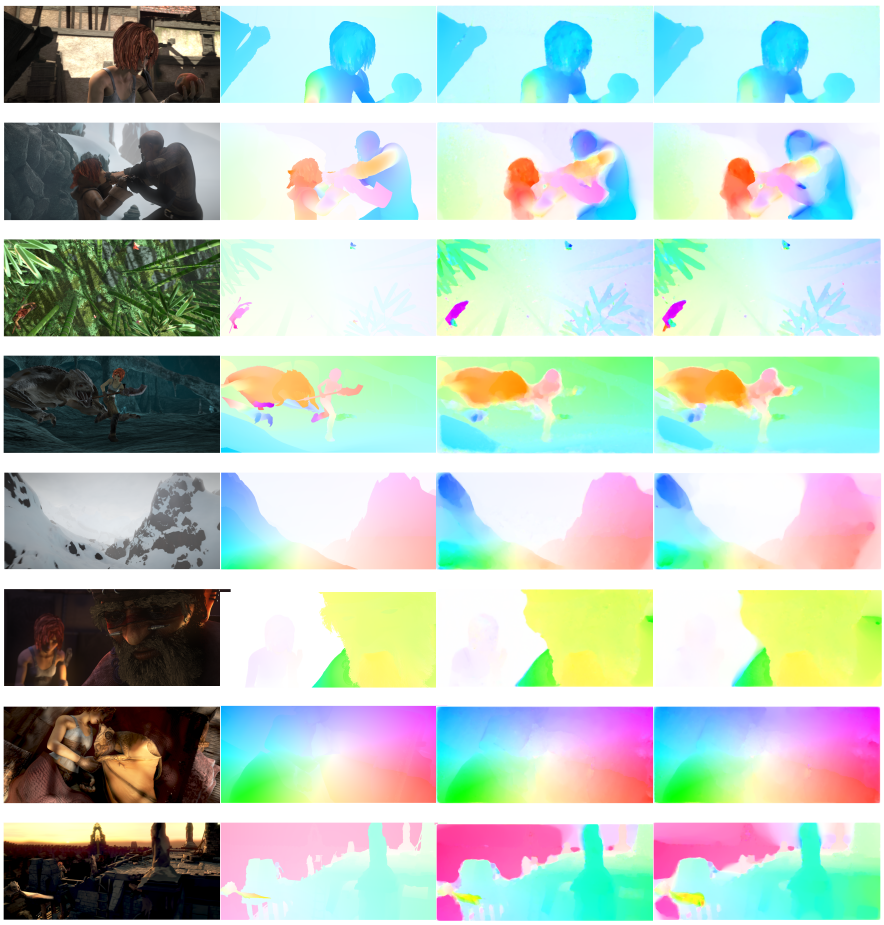


Fig. 5.11 Results on selected frames of MPI-Sintel. [150] Displacement field computed between the first frame (frame_0001) and the second (frame_0002)
Top to bottom rows: alley_1, ambush_5, bamboo_1, cave_2, mountain_1, shaman_2, sleeping_1, temple_2.
Left column to right: First frame (frame_0001), ground truth, clean, final.

5.2.8 Processing Time

The implementation of the code is done in MATLAB. The running time was not of concern at the time of implementing the algorithm. However, it is useful to get an idea of the running time for the algorithm. In general MATLAB codes may run slow in comparison to other languages such as C and C++, as a result it is expected that the algorithm runs slow. As part of future work a faster version can be implemented using a different programming language.

Table-5.8 depicts the running time of the algorithm to compute optical flow between two images. The code ran on a Dell laptop powered by an Intel Core i5 CPU with 2.40 MHZ clock speed.

Resolution	Time (grey-scale)	Time (colour)
380×420	$\simeq 283$ s.	$\simeq 457$ s.
388×584	$\simeq 376$ s.	$\simeq 738$ s.
480×640	$\simeq 488$ s.	$\simeq 860$ s.
436×1024	$\simeq 686$ s.	$\simeq 1297$ s.

Table 5.8 Running speed for different resolutions.

As expected the running time for colour image is much bigger than that for the grey-scale images, this is due to the three channels of colours requiring processing in the primal step.

5.2.9 Current Performance on test benchmarks

In addition to the training datasets that are available in evaluation benchmarks, test datasets are also available and are used to asses performance of methods. To this end the proposed algorithm is applied to the test dataset of the previously discussed datasets. In this subsection the results of these evaluations are demonstrated and discussed.

The Middlebury ranking

The Middlebury datasets include 8 image sequences that are used to asses algorithm performance. The proposed algorithm in this thesis was applied to the test images. Currently the algorithm has an average rank of (57.5) based on the AEPE values, and (57.8) based on the AAE values. Figure-5.12 next depicts a segments of the Middlebury ranking table for both measures.

Average angle error	avg. rank	Army (Hidden texture) GT im0 im1 all	disc	untext	Mequon (Hidden texture) GT im0 im1 all	disc	untext	Schefflera (Hidden texture) GT im0 im1 all	disc	untext	Wooden (Hidden texture) GT im0 im1 all	disc	untext	Grove (Synthetic) GT im0 im1 all	disc	untext	Urban (Synthetic) GT im0 im1 all	disc	untext	Yosemite (Synthetic) GT im0 im1 all	disc	untext	Teddy (Stereo) GT im0 im1 all	disc	untext
TCOF [69]	56.1	4.17 62	10.4 57	3.71 78	3.17 52	10.7 52	2.59 60	6.58 74	15.7 74	3.82 77	3.69 64	16.1 52	2.37 73	3.78 86	4.95 98	2.47 42	2.59 7	8.47 15	2.58 17	3.66 92	4.83 83	2.67 51	1.83 30	4.20 34	1.46 17
RFlow [90]	56.3	3.82 46	10.0 49	3.44 68	2.61 26	9.73 38	2.02 21	5.66 66	14.5 66	2.05 26	3.93 68	23.1 88	1.90 56	3.24 54	4.19 54	2.66 58	4.12 64	15.2 76	3.34 55	2.61 49	3.56 34	2.65 50	4.48 76	10.5 86	3.93 77
HBM-GC [106]	57.4	5.25 89	10.5 58	4.34 90	3.17 52	8.78 26	2.94 72	4.38 46	10.6 40	2.68 54	3.59 59	12.8 12	2.47 76	2.96 35	3.64 24	2.64 56	3.96 59	8.26 11	3.56 61	4.40 104	5.92 110	3.62 91	2.55 45	6.34 50	3.29 58
Steered-L1 [120]	57.8	3.30 31	8.44 23	2.91 42	**1.89** 1	7.14 8	1.60 6	3.61 30	9.91 35	1.89 19	3.45 54	19.4 67	1.64 44	3.42 67	4.30 63	3.39 80	5.18 82	14.5 69	4.37 83	5.09 112	5.05 90	10.1 115	5.56 89	10.2 81	6.24 97
ComplOF-FED-GPU [35]	59.2	4.28 70	11.3 68	3.70 77	3.25 59	13.0 75	2.16 28	4.06 42	11.2 45	1.95 21	3.91 67	19.2 65	2.01 60	3.20 49	4.15 51	2.64 56	4.61 72	16.1 82	3.90 70	2.98 71	3.77 46	3.69 92	2.85 51	7.44 59	2.53 44
SRR-TVOF-NL [91]	59.4	4.47 75	10.9 61	3.32 58	4.04 76	13.2 78	2.90 70	4.81 52	12.5 54	3.15 65	3.33 52	15.3 39	1.61 43	3.24 54	4.03 47	2.70 61	3.94 57	11.8 43	3.33 53	4.16 100	5.21 97	3.44 89	2.06 38	3.48 23	2.42 40
TF+OM [100]	61.3	3.97 52	10.2 52	2.94 44	2.91 42	9.12 32	2.57 59	5.22 60	11.5 46	6.92 88	3.59 59	16.1 52	2.28 70	3.20 49	3.97 41	3.11 73	4.70 76	14.5 69	4.32 81	3.06 74	4.84 84	2.71 53	3.93 67	8.79 67	4.32 82

Average endpoint error	avg. rank	Army (Hidden texture) GT im0 im1 all	disc	untext	Mequon (Hidden texture) GT im0 im1 all	disc	untext	Schefflera (Hidden texture) GT im0 im1 all	disc	untext	Wooden (Hidden texture) GT im0 im1 all	disc	untext	Grove (Synthetic) GT im0 im1 all	disc	untext	Urban (Synthetic) GT im0 im1 all	disc	untext	Yosemite (Synthetic) GT im0 im1 all	disc	untext	Teddy (Stereo) GT im0 im1 all	disc	untext
EpicFlow [103]	55.0	0.12 74	0.36 88	0.09 60	0.25 62	0.85 70	0.21 72	0.39 54	1.00 63	0.25 66	0.19 55	1.01 64	0.11 51	0.89 64	1.31 74	0.69 55	0.53 58	1.31 53	0.34 49	0.10 4	**0.11** 1	0.17 13	0.67 54	1.43 57	0.87 59
ComplOF-FED-GPU [35]	56.1	0.11 54	0.29 58	0.10 78	0.21 30	0.78 58	0.14 18	0.32 42	0.79 46	0.17 15	0.19 55	0.99 63	0.11 51	0.89 64	1.29 66	0.73 63	1.25 90	1.74 87	0.64 85	0.14 47	0.13 32	0.30 80	0.64 51	1.50 61	0.83 53
Classic++ [32]	57.4	0.09 30	0.25 36	0.07 15	0.23 51	0.78 58	0.19 53	0.43 60	1.00 63	0.22 57	0.20 61	1.11 72	0.10 46	0.87 57	1.30 68	0.66 53	0.47 42	1.62 76	0.33 43	0.17 81	0.14 54	0.32 90	0.79 75	1.64 71	0.92 65
Steered-L1 [120]	57.5	0.09 30	0.22 16	0.08 42	**0.14** 1	0.49 2	0.12 5	0.28 24	0.69 32	0.16 12	0.18 50	1.06 67	0.09 25	0.89 64	1.24 55	0.91 76	1.71 106	1.68 81	0.94 101	0.26 112	0.18 92	0.71 115	1.06 90	1.80 84	1.64 98
HBM-GC [106]	57.9	0.14 89	0.28 51	0.12 91	0.26 68	0.69 37	0.22 75	0.34 45	0.75 39	0.22 57	0.21 67	0.77 28	0.15 75	0.67 29	0.97 26	0.52 35	0.63 70	0.81 7	0.44 65	0.22 105	0.19 103	0.36 96	0.54 38	1.21 46	0.78 48
Aniso. Huber-L1 [22]	58.5	0.10 44	0.28 51	0.08 42	0.31 81	0.88 75	0.28 86	0.56 79	1.13 72	0.29 78	0.20 61	0.92 58	0.13 65	0.84 54	1.20 52	0.70 58	0.39 23	1.23 47	0.28 18	0.17 81	0.15 64	0.27 64	0.64 51	1.36 50	0.79 49
TF+OM [100]	59.3	0.10 44	0.26 42	0.07 15	0.22 37	0.66 30	0.19 53	0.36 49	0.78 44	0.39 86	0.20 61	0.89 53	0.13 65	0.98 82	1.31 74	1.03 85	0.56 62	1.55 73	0.33 43	0.16 72	0.17 82	0.27 64	0.76 67	1.59 69	0.98 72

Fig. 5.12 Middleburry ranking table.
Top: AAE results ranking. Bottom: AEPE results ranking.

To further examine and compare the results with other algorithms, the values of AEPE are listed in the Table-5.9 with the results of other method sharing similar principles. These methods are, the Improved-TV-L^1 [14], CLG-TV [4], LDOF [13].

Image	LDOF [13]	CLG-TV [4]	Improved-$TV-L^1$ [14]	Steered-L^1
Army	$0.12_{(74)}$	$0.11_{(54)}$	$0.09_{(30)}$	$0.09_{(30)}$
Mequon	$0.23_{(84)}$	$0.32_{(84)}$	$0.20_{(28)}$	$0.14_{(1)}$
Schefflera	$0.43_{(60)}$	$0.55_{(77)}$	$0.53_{(73)}$	$0.28_{(24)}$
Wooden	$0.45_{(98)}$	$0.25_{(78)}$	$0.21_{(67)}$	$0.18_{(50)}$
Grove	$1.01_{(86)}$	$0.92_{(71)}$	$0.90_{(67)}$	$0.89_{(64)}$
Urban	$1.10_{(86)}$	$0.47_{(42)}$	$1.51_{(101)}$	$1.71_{(106)}$
Yosemite	$0.12_{(27)}$	$0.17_{(81)}$	$0.18_{(88)}$	$0.26_{(112)}$
Teddy	$0.94_{(86)}$	$0.74_{(65)}$	$0.73_{(62)}$	$1.06_{(90)}$
Average rank	80.5	69.5	63.8	57.5

Table 5.9 AEPE comparison for four algorithms taken from the Middlebury ranking table. Numbers in brackets indicate the ranking of the specific image sequence results, for example the results of the 'Mequon' sequence of the algorithm proposed in this thesis is ranked first. Numbers in blue indicate the highest rank in this table.

This table illustrates the AEPE values and the rank on the Middlebury ranking list. The rank in this table is illustrated using two numbers, for example the Steered-L^1 scored $0.14_{(1)}$ in the 'Mequon' image sequence. The value 0.14 represents the AEPE results (Section-3.4.2), which is calculated as the difference between the obtained flow field and the ground truth (Equation-3.24), the smaller the number the closer the estimated flow field to the ground truth. The number between the brackets '(1)' depicts the rank on the particular method (steered-L^1 in this example), for the designated image sequence ('Mequon' in this example). The 'Average rank' for each algorithm is the average of ranks for all the eight image sequences. Figure-5.13 depicts several example of the test images along with their estimated flow field

Fig. 5.13 Optical flow visualisation of some Middlebury test images obtained via steered-L^1.
left: frame_10. Right: Visualised optical flow field.
Image sequences top to bottom: 'Mequon' ranked 1st. according to its AEPE. 'Schefflera' ranked 24th. according to its AEPE. 'Wooden' ranked 50th. according to its AEPE.

MPI-Sintel Ranking

MPI-Sintel contain a huge number of test image sequences. Test image sequences belong to the 'clean' and 'final' passes. As discussed earlier the MPI-Sintel is a very challenging dataset. This is due to the large and complex motion it contains, varied textures and illumination effects. Methods with high-ranking on the Middlebury dataset have more difficulty estimating optical flow on this dataset [150]. The AEPE for the test were obtained by applying the proposed algorithm here. The AEPE for the test images in the 'clean' pass is equal to (10.864), and for'final' pass is equal to (12.277).

Despite that improved performance that this algorithm produces, it still has some limitations in several aspects. The algorithm improved the robustness to illumination as was demonstrated earlier; however it still suffers in the presence of bigger illumination changes, in addition to that the algorithm performance degrades at the presence of large motion and occlusion. This can be noticed by the relative lower accuracy on the MPI-Sintel dataset. Another issue related to the piecewise nature of the L^1 norm. As discussed earlier, the L^1 norm has a piecewise behaviour, which results in favouring piecewise solutions of the optical flow field. This can be noticed in areas with slightly slanted motion [95], in this case the estimated flow appears to have piecewise constant flow field rather than a smooth flow field. On the computation time level, the implemented algorithm is not suitable for real-time performance as it requires high computation time. These limitations are further discussed in Chapter-6 with possible future work suggestions.

5.3 Summary

In this chapter several experiments were conducted to investigates the algorithms proposed in this thesis. This chapter is divided into two parts. The first is dedicated to experiments conducted on the CLG optical flow image registration. The experiments highlighted the improved performance of image registration with the use of CLG optical flow. In this part the performance of the image registration was visually inspected and compared with the performance of the global optical flow algorithm of Horn-Schunck [1]. Additionally, experiments were used to investigate an improved CLG optical flow algorithm formulated earlier in this thesis (see Section 3.8).

In the second part of this chapter, the proposed steered-L^1 was investigated. Several experiments were conducted to test the effect that the main constituents of the algorithm have on the algorithm performance. Experiments were conducted mainly on the Middlebury dataset [6], some tests were also done using the MPI-Sintel dataset [150]. The proposed algorithm improves the $TV-L^1$ algorithm [22], [14] by improving the filling-in effect along motion edges. Despite not being the best preforming method at the current time, but the proposed algorithm can be further improved. This is going to be discussed in more details in the next chapter. It was shown also in this chapter that this algorithm has a higher ranking on the Middlebury benchmark compared with algorithms sharing similar principals.

Chapter 6

Discussion, Conclusion and Future Work

In this thesis, the topic of optical flow estimation was investigated. Several algorithms were explored in relation to a number of applications. In Chapter-2 a literature review for optical flow algorithms was presented. The chapter focused mainly on the variational methods for finding optical flow. The main problems that faced the estimation of optical flow were also investigated along with researchers solutions for such issues. The advances of variational methods and techniques were explored. Additionally this chapter included a discussion on the main applications in which optical flow estimation is used. It was shown that this low-level image processing technique can have a direct application in several computer vision problems.

In Chapter-3 a new image registration method was proposed that is based on the CLG optical flow calculation. In addition to that a new version of CLG optical flow was presented, which replaces the Gaussian filter with a non-linear bi-lateral filter. It was shown that the performance of image registration using this method surpasses that of the global method of Horn-Schunck [1], especially in terms of preserving edges. Chapter-3 can also be considered as an introduction to the concept of optical flow. Theoretical notions discussed in that chapter are the fundamental ideas underpinning the majority of variational optical flow algorithms.

Chapter-4 introduced the main contribution of this thesis, the steered-L^1 norm for optical flow computation. The L^1 smoothness term suffers from two major drawbacks. First this function is not continuously differentiable, second this term is isotropic. The first issue was addressed by Zach et al. [22] with a dual formulation numerical scheme [91], [162]. This algorithm however still suffers from being isotropic. The proposed algorithm in Chapter-4 can be considered as an anisotropic version of the L^1 smoothness term previously discussed

in [22]. In addition to that, a new formulation of the data term was introduced which is inspired by the non-linearised robust data term proposed by Brox et al. [2]. The data term also incorporates an images gradients term to improve the robustness against illumination changes.

In Chapter-5 experiments were conducted to assess the performance of the proposed algorithms, and to compare the efficiency of the algorithms with several algorithms which follow a similar approach for estimating optical flow. This chapter is divided into two parts, the first part is dedicated to experiments highlighting image registration performance using the proposed method. This section also includes a comparison with image registration using the global method of Horn-Schunck [1]. Middlebury dataset image sequences are used in these experiments. In the second part of this chapter, several experiments are performed to investigate the performance of the proposed optical flow estimation method. Each experiment highlighted the improvement for different contributing constituents of the algorithm.

Contribution

The Contributions of this thesis can be summarised in the following points:

- Proposing a new method for image registration based on the CLG optical flow algorithm;
- Proposing an improved CLG optical flow based on the use of bi-lateral filtering;
- Proposing a steered-L^1 norm smoothness term, which enabled an improved filling-in effect in the estimated flow field;
- Deriving data term formulation for primal-dual optical flow computation based on the non-linearised optical flow algorithm;
- Introduction of new extended intermediate filtering

6.1 Discussion and Future Work

The following discussion is mainly dedicated to assess the performance of the optical flow algorithm with the steered-L^1 norm that was proposed in Chapter-4. It is acknowledged here that the results of the experiments are not the best performing compared to the state-of-the-art optical flow algorithms on their own. However, the contribution here can be an important addition to the building block of variational optical flow, especially algorithms

following the primal-dual minimisation. Further experiments can be conducted to investigate the performance of the proposed algorithm using different datasets. As discussed earlier several datasets are available and can be used to evaluate optical flow algorithms, such as the KITTI benchmark [151], and the MIP-Sintel [150]. The algorithm was evaluated using the Middlebury and the MPI-Sintel datasets, it is also proposed that the algorithm be evaluated via the KITTI optical flow assessment benchmark in the future. The KITTI dataset contains more specific images aimed at autonomous driving (Section-3.4.1). The KITTI dataset was not used as it offers ground truth with only 50% density.

Further research should be directed towards the improvement of the method proposed in this thesis. In this thesis it was shown that the steered-L^1 norm can improve the accuracy of optical flow estimation. Several aspects of the algorithm can be investigated to improve the performance and to make this algorithm competitive with the state-of-the-art algorithms, these are discussed in the following points.

- **Other smoothness terms:** The performance of the proposed algorithm can be further improved using some additional techniques borrowed from already existing algorithms. As discussed earlier, the L^1 norm favours a piecewise constant solution, resulting in a stair-casing effect in smooth areas. This problem can be addressed by using higher order norms [2]. Alternatively other penalisers can be used, such as the Huber-L^1 norm [163]. This penaliser behaves as a quadratic penaliser for relatively smooth areas, and as L^1 norm in areas with higher gradients. Werlberger et al. [23] proposed an anisotropic image-driven Huber-L^1 to improve the performance of such a penaliser. For future work, a steered anisotropic Huber-L^1 could be designed to further improve optical flow estimation.
- **Different Colour Spaces:** In order to improve the illumination robustness of the proposed algorithm, an image gradients term is used to increase the robustness of the data term. This term is added to cope with the illumination changes between images. Another idea that can improve the performance is the use of different colour spaces [72]. Colour spaces such as the HSV colour space were found to improve the illumination robustness in some algorithms [15], [67]. The HSV components are the Hue, Saturation, and Value. The hue channel is robust under multiplicative illumination especially shadow, shading, highlights and specularities [15]. In the current thesis grey-scale and colour images are used for experiments, The colour images belong to the RGB colour space. For future work experiments are to be conducted to assess the performance of images of HSV colours. It is expected to improve the estimation of the displacement flow field significantly [15], [67].

- **Spatio-temporal smoothness term:** The estimation of optical flow in the current thesis rely on the spatial continuity assumption, in other words estimates optical flow between two images only. In reality such images may be a part of a long sequence of images (e.g. video sequence). Spatio-temporal constraints were used in some algorithms to improve the performance of optical flow algorithms in case long image sequences are available [65], [103]. In spatio-temporal smoothness terms it is assumed that the flow changes gradually over time, and that previous as well as the next frame will demonstrate relatively similar displacements. Generally the use of spatio-temporal smoothness terms were more exploited in the case of Euler-Lagrange minimisation rather than the algorithms following primal-dual minimisation [23]. Werlberger et al. [23] argues that the spatio-temporal smoothness terms were not very successful when tested using the Middlebury dataset. The reason as the authors suggest is that the Middlebury datasets contain more complex displacements in comparison to some previous benchmarks such as the Yosemite [1] benchmark. Alternatively the authors in [23] propose a new method for temporal smoothness that relies on *'mirroring'* of the data constancy term. Generally in the literature there is a lack of investigation of spatio-temporal smoothness terms for the primal-dual algorithms. As future work spatio-temporal terms could be examined for primal-dual algorithms. The spatio-temporal and the temporal data terms [23] could both be investigated to improve the performance of the current algorithm.
- **Implementation:** Optical flow algorithms working in the primal-dual settings usually belong to the high performing algorithms in terms of computation time. Such algorithms can be easily parallelised using modern GPUs [38]. The proposed algorithm here has several differences when compared to previously proposed primal-dual algorithms which were successfully parallelised and found to achieve high speed performance [22], [14], [4]. One of these differences is the delay of data term linearisation as was proposed in [2], in addition to some extra calculation in the smoothness term due to the addition of the structure tensor and eigenvectors computation. The implementation of the algorithm in this thesis was done using MATLAB. As a future plan, an implementation of this algorithm should be considered in C++. Several libraries may be considered in the future implementation including OpenCV. Vectorisation using the Intel compiler should also be considered [35]. Further it would be useful to explore the possibility of a parallel version of the code on a GPU using the NVidia CUDA platform. The aim of this implementation would be to exploit the algorithm and investigate whether real-time or near real-time performance is possible. This

[1] http://cs.brown.edu/people/black/images.html

would be useful if this algorithm is to be used in real-time applications (e.g. mobile robot navigation)

- **Applications and extension to LDOF :** As pointed out earlier, the estimation of optical flow has numerous applications in image processing and computer vision fields. Application for optical flow were reviewed and discussed earlier in this thesis in Section-2.3. The proposed algorithm in this thesis can be further extended in a similar way to the LDOF algorithm [13]. This can be done by incorporating a descriptors matching function in the data term. A possible research direction is the use of optical flow in mobile robots obstacle detection, and also visually impaired navigation assistant. Obstacle in the path of robots can have different sizes. While optical flow can be used to detect obstacles and steer robots away from it, the inherited shortcoming of failure to detect the motion of small objects with large displacements is hazardous. Therefore, and in order to utilise this algorithm in any navigation system, it has to be able to find displacements even for small objects. Some preliminary research has already been done in this aspect, which can be found in Appendix-A.

References

[1] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981. pages 1, 7, 9, 12, 15, 19, 20, 26, 28, 35, 39, 71, 72, 73, 77, 79, 81, 84, 94, 102, 103, 105, 130, 131, 132, 149, 151, 156

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," May 2004. pages 1, 15, 20, 21, 22, 23, 28, 81, 84, 88, 94, 99, 101, 123, 132, 133, 134, 149

[3] A. Bruhn and J. Weickert, "Towards ultimate motion estimation: Combining highest accuracy with real-time performance," in *10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China*, pp. 749–755, 2005. pages 1, 14, 15

[4] M. Drulea and S. Nedevschi, "Total variation regularization of local-global optical flow," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pp. 318–323, Oct 2011. pages 1, 10, 14, 20, 26, 62, 77, 82, 89, 92, 94, 128, 134

[5] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *InternationalL Journal of Computer Vision*, vol. 12, pp. 43–77, 1994. pages 2, 13, 69

[6] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vision*, vol. 92, pp. 1–31, Mar. 2011. pages 2, 3, 4, 7, 12, 52, 54, 59, 65, 66, 67, 69, 70, 102, 103, 104, 112, 114, 124, 130

[7] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable Medical Image Registration: A Survey," *Medical Imaging, IEEE Transactions on*, vol. 32, pp. 1153–1190, July 2013. pages 6, 36, 39

[8] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, pp. 977–1000, 2003. pages 6, 39, 50, 51

[9] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof, "A duality based algorithm for tv-l1-optical-flow image registration," *Med Image Comput Comput Assist Interv*, vol. 10, no. Pt 2, pp. 511–518, 2007. pages 6, 8, 10, 36, 89

[10] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pp. 674–679, April 1981. pages 6, 8, 15, 19, 30, 35, 39, 65, 71, 73

[11] D. Sun, S. Roth, J. Lewis, and M. J. Black, "Learning optical flow," in *European Conf. on Computer Vision, ECCV* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5304 of *LNCS*, p. 83–97, Springer-Verlag, Oct. 2008. pages 7, 9, 17, 26, 30, 31, 52, 81, 90, 92

[12] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International Journal of Computer Vision*, vol. 61, pp. 211–231, 2005. pages 7, 8, 11, 14, 15, 19, 20, 21, 28, 39, 45, 63, 71, 73, 74, 75, 77, 84, 86, 108, 109

[13] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011. pages 8, 14, 16, 21, 24, 36, 53, 84, 88, 123, 124, 128, 135, 149, 150, 151, 152, 153, 156

[14] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers, "Statistical and geometrical approaches to visual motion analysis," ch. An Improved Algorithm for TV-L1 Optical Flow, pp. 23–45, Berlin, Heidelberg: Springer-Verlag, 2009. pages 10, 14, 16, 20, 36, 59, 94, 99, 114, 123, 124, 128, 130, 134

[15] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, and H. Seidel, "Complementary optic flow," in *Proceedings of the 7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR '09, (Berlin, Heidelberg), pp. 207–220, Springer-Verlag, 2009. pages 10, 16, 23, 30, 81, 90, 92, 101, 102, 133

[16] P. Anandan, "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision*, vol. 2, no. 3, pp. 283–310, 1989. pages 14

[17] B. Kitt, B. Ranft, and H. Lategahn, "Block-matching based optical flow estimation with reduced search space based on geometric constraints," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 1104–1109, Sept 2010. pages 14

[18] J. Xuan and L.-P. Chau, "An efficient three-step search algorithm for block motion estimation," *Multimedia, IEEE Transactions on*, vol. 6, pp. 435–438, June 2004. pages 14

[19] D. J. Heeger, "Model for the extraction of image flow," *J. Opt. Soc. Am. A*, 1987. pages 14

[20] 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. pages 14, 69

[21] K. Pauwels and M. V. Hulle, "Realtime phase-based optical flow on the gpu," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pp. 1–8, June 2008. pages 14

[22] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Proceedings of the 29th DAGM conference on Pattern recognition*, (Berlin, Heidelberg), pp. 214–223, Springer-Verlag, 2007. pages 14, 16, 20, 25, 26, 27, 36, 80, 81, 82, 89, 91, 109, 114, 130, 131, 132, 134

[23] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof, "Anisotropic Huber-L1 optical flow," in *Proceedings of the British Machine Vision Conference (BMVC)*, (London, UK), September 2009. pages 14, 16, 26, 27, 82, 89, 119, 133, 134

[24] L. Hoeltgen, S. Setzer, and M. Breuß, "Intermediate flow field filtering in energy based optic flow computations.," in *EMMCVPR* (Y. Boykov, F. Kahl, V. S. Lempitsky, and F. R. Schmidt, eds.), vol. 6819 of *Lecture Notes in Computer Science*, pp. 315–328, Springer, 2011. pages 14, 59, 119

[25] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr, "Variational optical flow computation in real time," *Image Processing, IEEE Transactions on*, vol. 14, pp. 608–615, Apr. 2005. pages 15

[26] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr, "Real-time optic flow computation with variational methods.," in *CAIP* (N. Petkov and M. A. Westenberg, eds.), vol. 2756 of *Lecture Notes in Computer Science*, pp. 222–229, Springer, 2003. pages 15

[27] W. L. Briggs, V. E. Henson, and M. S. F., *A Multigrid Tutorial: Second Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. pages 15

[28] E. Kalmoun, H. Köstler, and U. Rüde, "3d optical flow computation using a parallel variational multigrid scheme with application to cardiac c-arm ct motion," *Image Vision Comput.*, vol. 25, pp. 1482–1494, Sept. 2007. pages 15

[29] P. Gwosdek, H. Zimmer, S. Grewenig, A. Bruhn, and J. Weickert, "A highly efficient gpu implementation for variational optic flow based on the euler-lagrange framework," in *Trends and Topics in Computer Vision* (K. Kutulakos, ed.), vol. 6554 of *Lecture Notes in Computer Science*, pp. 372–383, Springer Berlin Heidelberg, 2012. pages 15, 16

[30] J. Marzat, Y. Dumortier, and A. Ducrot, "Real-time dense and accurate parallel optical flow using cuda," in *Proceedings of the 17th International Conference WSCG, Plzen, Czech Republic*, Feb 2009. pages 15

[31] A. Plyer, G. L. Besnerais, and F. Champagnat, "Massively parallel lucas kanade optical flow for real-time video processing applications," *Journal of Real- Time Image Processing*, pp. 1–18, Apr. 2014. pages 15

[32] V. Mahalingam, K. Bhattacharya, N. Ranganathan, H. Chakravarthula, R. R. Murphy, and K. S. Pratt, "A vlsi architecture and algorithm for lucas-kanade-based optical flow computation.," *IEEE Trans. VLSI Syst.*, vol. 18, no. 1, pp. 29–38, 2010. pages 15

[33] J. Diaz, E. Ros, F. Pelayo, E. Ortigosa, and S. Mota, "Fpga-based real-time optical-flow system," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, pp. 274–279, Feb 2006. pages 15

[34] S. Grewenig, J. Weickert, and A. Bruhn, "From box filtering to fast explicit diffusion," in *Proceedings of the 32Nd DAGM Conference on Pattern Recognition*, (Berlin, Heidelberg), pp. 533–542, Springer-Verlag, 2010. pages 16

[35] N. Sundaram, T. Brox, and K. Keutzer, "Dense point trajectories by gpu-accelerated large displacement optical flow," Sept. 2010. pages 16, 36, 134

[36] J. Shi and C. Tomasi, "Good features to track," in *1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pp. 593 – 600, 1994. pages 16

[37] C. Zach, D. Gallup, and J. M. Frahm, "Fast gain-adaptive KLT tracking on the GPU," in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition Workshops CVPRW '08*, 2008. pages 16

[38] M. Werlberger, T. Pock, and H. Bischof, "Motion estimation with non-local total variation regularization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, (San Francisco, CA, USA), June 2010. pages 16, 31, 89, 134, 156

[39] D. Fortun, P. Bouthemy, and C. Kervrann, "Optical flow modeling and computation: a survey," *Computer Vision and Image Understanding*, vol. 134, p. 21, May 2015. pages 16, 17

[40] S. Roth and M. J. Black, "Steerable random fields," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Oct 2007. pages 17, 30, 64, 90, 110

[41] S. Roth and M. J. Black, "On the spatial statistics of optical flow," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 33–50, 2007. pages 17

[42] J. Vlontzos and D. Geiger, "A mrf approach to optical flow estimation," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on*, pp. 853–856, Jun 1992. pages 17

[43] F. Heitz and P. Bouthemy, "Multimodal estimation of discontinuous optical flow using markov random fields," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, pp. 1217–1232, Dec 1993. pages 17, 18

[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *SCIENCE*, vol. 220, no. 4598, pp. 671–680, 1983. pages 17

[45] B. Buxton, A. Kashko, and H. Buxton, "Optical flow matching by simulated annealing," tech. rep., GEC Hirst Research Centre and Department of Computer Science, 1990. pages 17

[46] Y. Boykov, O. V. R., and Zabih, "Markov random fields with efficient approximations," in *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pp. 648–655, Jun 1998. pages 17

[47] Y. Boykov, O. V. R., and Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 1222–1239, Nov. 2001. pages 17, 25

[48] Y. Boykov and O. Veksler, "Graph cuts in vision and graphics: Theories and applications," in *Handbook of Mathematical Models in Computer Vision* (N. Paragios, Y. Chen, and O. Faugeras, eds.), pp. 79–96, Springer US, 2006. pages 18

[49] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. pages 18

[50] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. pages 18

[51] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *CoRR*, vol. abs/1406.2283, 2014. pages 18

[52] V. Lempitsky, S. Roth, and C. Rother, "Fusionflow: Discrete-continuous optimization for optical flow estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE Computer Society, June 2008. pages 18

[53] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu, "Large displacement optical flow from nearest neighbor fields," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. pages 18, 24, 150

[54] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow," in *Computer Vision and Pattern Recognition*, 2015. pages 18, 25

[55] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *ICCV 2013 - IEEE Intenational Conference on Computer Vision*, (Sydney, Australia), pp. 1385–1392, IEEE, Dec. 2013. pages 18, 25

[56] M. Okutomi and T. Kanade, "A locally adaptive window for signal matching," in *Computer Vision, 1990. Proceedings, Third International Conference on*, pp. 190–199, Dec 1990. pages 19

[57] X. Ren, "Local grouping for optical flow," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008. pages 19

[58] T. K. Leung and J. Malik, "Contour continuity in region based image segmentation.," in *ECCV (1)* (H. Burkhardt and B. Neumann, eds.), vol. 1406 of *Lecture Notes in Computer Science*, pp. 544–559, Springer, 1998. pages 19

[59] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, (Washington, DC, USA), pp. 839–, IEEE Computer Society, 1998. pages 20, 29, 60, 77, 119

[60] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, (New York, NY, USA), pp. 257–266, ACM, 2002. pages 20, 60, 61, 77, 119, 157

[61] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert, "Highly accurate optic flow computation with theoretically justified warping," *Int. J. Comput. Vision*, vol. 67, pp. 141–158, Apr. 2006. pages 20, 21, 23, 28, 88

[62] Y. Kim, A. Martínez, and A. Kak, "Robust motion estimation under varying illumination," *Image Vision Comput.*, vol. 23, pp. 365–375, Apr. 2005. pages 20, 22

[63] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, Nov. 1992. pages 20, 81

[64] M. J. Blackand and P. Anandan, "A framework for the robust estimation of optical flow," in *IEEE ICCV*, pp. 231–236, 1993. pages 21, 27, 28, 29, 84, 85

[65] M. J. Black and P. Anandan, "Robust dynamic motion estimation over time," in *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pp. 296–302, Jun 1991. pages 21, 28, 29, 30, 85, 134

[66] M. J. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Comput. Vis. Image Underst.*, vol. 63, pp. 75–104, Jan. 1996. pages 21, 28, 29, 84, 85

[67] H. Zimmer, A. Bruhn, and J. Weickert, "Optic flow in harmony," *Int. Journal Comput. Vision*, vol. 93, pp. 368–388, July 2011. pages 21, 22, 30, 90, 92, 133, 155

[68] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1744–1757, 2012. pages 21, 24

[69] D. Geiger and F. Girosi, "Parallel and deterministic algorithms from mrfs: surface reconstruction and integration," in *Computer Vision - ECCV'90, First European Conference on Computer Vision, Antibes, France, April 23-27, 1990, Proceedings*, pp. 89–98, 1990. pages 21

[70] S. Negahdaripour, "Revised definition of optical flow: integration of radiometric and geometric cues for dynamic scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, pp. 961–979, Sep 1998. pages 22

[71] J. van de Weijer and T. Gevers, "Robust optical flow from photometric invariants," in *Image Processing, 2004. ICIP '04. 2004 International Conference on*, vol. 3, pp. 1835–1838 Vol. 3, 2004. pages 22

[72] Y. Mileva, A. Bruhn, and J. Weickert, "Illumination-robust variational optical flow with photometric invariants," in *In DAGM-Symposium, LNCS 4713*, pp. 152–162, 2007. pages 22, 99, 133

[73] R. J. Andrews and B. C. Lovell, "Color optical flow," in *Proceedings Workshop on Digital Image Computing*, pp. 135–139, 2003. pages 22, 99

[74] J. Barron and R. Klette, "Quantitative color optical flow," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, pp. 251–255 vol.4, 2002. pages 22

[75] K. R. T. Aires, A. M. Santana, and A. A. D. Medeiros, "Optical flow using color information: Preliminary results," in *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, (New York, NY, USA), pp. 1607–1611, ACM, 2008. pages 22

[76] S.-H. Lai and B. Vemuri, "Reliable and efficient computation of optical flow," *International Journal of Computer Vision*, vol. 29, no. 2, pp. 87–105, 1998. pages 22, 23

[77] P. Golland and A. M. Bruckstein, "Motion from color.," *Computer Vision and Image Understanding*, vol. 68, no. 3, pp. 346–362, 1997. pages 22

[78] F. Steinbrücker, T. Pock, and D. Cremers, "Advanced data terms for variational optic flow estimation," in *VMV*, pp. 155–164, 2009. pages 22

[79] F. Steinbrücker, T. Pock, and D. Cremers, "Large displacement optical flow computation without warping," in *ICCV*, pp. 1609–1614, 2009. pages 22, 25, 36, 150

[80] L. Álvarez, J. Weickert, and J. Sánchez, "Reliable estimation of dense optical flow fields with large displacements.," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 41–56, 2000. pages 23, 29

[81] L. Alvarez, J. Esclarin, M. Lefebure, and J. sanchez, "A pde model for computing the optical flow," in *Proc. XVI congreso de ecuaciones diferenciales y aplicaciones*, (Las Palmas de Gran Canaria, Spain), pp. 1349–1356, 1999. pages 23, 28

[82] H. H. Nagel and W. Enkelmann, "An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 565–593, May 1986. pages 23

[83] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition* (C. Schmid, S. Soatto, and C. Tomasi, eds.), vol. 2, (INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334), pp. 886–893, June 2005. pages 24, 151, 152

[84] T. Brox, C. Bregler, and J. Malik, "Large displacement optical flow," in *CVPR*, pp. 41–48, 2009. pages 24, 88, 149

[85] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, pp. 91–110, Nov. 2004. pages 24, 152

[86] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," in *CVPR*, pp. 1293–1300, 2010. pages 24

[87] C. Rother, V. Kolmogorov, V. S. Lempitsky, and M. Szummer, "Optimizing binary mrfs via extended roof duality," in *CVPR*, 2007. pages 24

[88] S. Korman and S. Avidan, "Coherency sensitive hashing," in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, (Washington, DC, USA), pp. 1607–1614, IEEE Computer Society, 2011. pages 24

[89] L. Bao, Q. Yang, and H. Jin, "Fast edge-preserving patchmatch for large displacement optical flow," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2014. pages 25, 124, 150

[90] M. Leordeanu, A. Zanfir, and C. Sminchisescu, "Locally affine sparse-to-dense matching for motion and occlusion estimation," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2013. pages 25, 115, 120

[91] A. Chambolle, "An algorithm for total variation minimization and applications," *J. Math. Imaging Vis.*, vol. 20, pp. 89–97, Jan. 2004. pages 26, 81, 82, 83, 131

[92] A. Wedel, D. Cremers, T. Pock, and H. Bischof, "Structure- and motion-adaptive regularization for high accuracy optic flow," (Kyoto, Japan), 2009. pages 26

[93] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 629–639, July 1990. pages 26, 90

[94] J. Weickert and C. Schnörr, "A theoretical framework for convex regularizers in pde-based computation of image motion," *International Journal of Computer Vision*, vol. 45, no. 3, pp. 245–264, 2001. pages 27, 72, 84

[95] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Img. Sci.*, vol. 3, pp. 492–526, Sept. 2010. pages 27, 130

[96] A. Buades, B. Coll, and J. Morel, "The staircasing effect in neighborhood filters and its solution," *IEEE Transactions on Image Processing*, vol. 15, no. 6, pp. 1499–1505, 2006. pages 27, 119

[97] W. Trobin, T. Pock, D. Cremers, and H. Bischof, "An unbiased second-order prior for high-accuracy motion estimation.," in *DAGM-Symposium* (G. Rigoll, ed.), vol. 5096 of *Lecture Notes in Computer Science*, pp. 396–405, Springer, 2008. pages 27

[98] J. Braux-Zin, R. Dupont, and A. Bartoli, "A general dense image matching framework combining direct and feature-based costs," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 185–192, Dec 2013. pages 27, 94, 123, 124, 150

[99] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pp. 467–474, Nov 2011. pages 27, 94, 123

[100] J. Weickert, "On discontinuity-preserving optic flow," in *In Proc. Computer Vision and Mobile Robotics Workshop*, pp. 115–122, 1998. pages 28, 29

[101] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," in *Image Processing, 1994. Proceedings. ICIP-94., IEEE Int. Conf.*, vol. 2, pp. 168 –172 vol.2, 1994. pages 28, 29

[102] G. Aubert, R. Deriche, and P. Kornprobst, "Computing optical flow via variational techniques," *SIAM Journal on Applied Mathematics*, vol. 60, no. 1, pp. 156–182, 1999. pages 28

[103] J. Weickert and C. Schnörr, "Variational Optic Flow Computation with a Spatio-Temporal Smoothness Constraint," *J. Math. Imaging and Vision*, vol. 14, no. 3, pp. 245–255, 2001. pages 28, 30, 134

[104] J. Weickert and C. Schnörr, "A Theoretical Framework for Convex Regularizers in PDE–Based Computation of Image Motion," *Int. J. Computer Vision*, vol. 45, no. 3, pp. 245–264, 2001. pages 28

[105] H. hellmut Nagel, "Constraints for the estimation of displacement vector fields from image sequences," in *In International Joint Conference on Artificial Intelligence*, pp. 945–951, 1983. pages 29

[106] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering-based optical flow estimation with occlusion detection.," in *ECCV (1)* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 211–224, Springer, 2006. pages 29, 62

[107] G. Farnebäck, "Fast and accurate motion estimation using orientation tensors and parametric motion models," in *Proceedings of 15th International Conference on Pattern Recognition*, vol. 1, (Barcelona, Spain), pp. 135–139, IAPR, September 2000. pages 30

[108] G. Farnebäck, "Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field," in *Proceedings of the Eighth IEEE International Conference on Computer Vision*, vol. I, (Vancouver, Canada), pp. 171–177, July 2001. pages 30

[109] P. Krähenbühl and V. Koltun, "Efficient nonlocal regularization for optical flow," in *Computer Vision – ECCV 2012* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7572 of *Lecture Notes in Computer Science*, pp. 356–369, Springer Berlin Heidelberg, 2012. pages 31

[110] A. Flynn, "Combining sonar and infrared sensors for mobile robot navigation," *The International Journal of Robotics Research*, vol. 7, no. 6, pp. 5–14, 1988. pages 32

[111] R. Carelli and E. O. Freire, "Corridor navigation and wall-following stable control for sonar-based mobile robots," *Robotics and Autonomous Systems*, vol. 45, no. 3–4, pp. 235 – 247, 2003. pages 32

[112] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *J. Intell. Robotics Syst.*, vol. 53, pp. 263–296, Nov. 2008. pages 32, 33

[113] M. V. Srinivasan, J. S. Chahl, K. Weber, S. Venkatesh, M. G. Nagle, and S. W. Zhang, "Robot navigation inspired by principles of insect vision," *Robotics and Autonomous Systems*, vol. 26, no. 2-3, pp. 203–216, 1999. pages 32, 33, 150, 157

[114] G. Desouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 237–267, Feb 2002. pages 32

[115] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi, "Divergent stereo for robot navigation: learning from bees," in *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pp. 434–439, Jun 1993. pages 32, 150, 157

[116] M. G. Nagle., M. V. Srinivasan, and P. J. Sobey, "Robust depth extraction for mobile robots," 1993. pages 33

[117] M. Srinivasan, M. Lehrer, S. Zhang, and G. Horridge, "How honeybees measure their distance from objects of unknown size," *Journal of Comparative Physiology A*, vol. 165, no. 5, pp. 605–613, 1989. pages 33

[118] T. Camus, D. Coombs, M. Herman, and T. Hong, "Real-time single-workstation obstacle avoidance using only wide-field flow divergence," in *Journal of Computer Vision Research*, pp. 323–330, 1996. pages 33, 34

[119] T. Camus, "Real-time quantized optical flow," in *Computer Architectures for Machine Perception, 1995. Proceedings. CAMP '95*, pp. 126–131, Sep 1995. pages 33

[120] R. C. Nelson and J. Aloimonos, "Obstacle avoidance using flow field divergence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 11, pp. 1102–1106, Oct 1989. pages 34

[121] K. Souhila and A. Karim, "Optical flow based robot obstacle avoidance," *International Journal of Advanced Robotic Systems*, 2007. pages 34

[122] T. Low and G. Wyeth, "Obstacle detection using optical flow," in *Australasian Conference on Robotics and Automation 2005* (C. Sammut, ed.), (Sydney, N.S.W), Australian Robotics and Automation Association Inc, 2005. pages 35

[123] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988. pages 35

[124] N. Ohnishi and A. Imiya, "Dominant plane detection from optical flow for robot navigation," *Pattern Recognition Letters*, vol. 27, no. 9, pp. 1009 – 1021, 2006. pages 35, 81

[125] A. Engelsberg and G. Schmidt, "A comparative review of digital image stabilising algorithms for mobile video communications," in *Consumer Electronics, 1999. ICCE. International Conference on*, pp. 88–89, June 1999. pages 35

[126] J. Chang, W. Hu, M. Cheng, and B. Chang, "Digital image translational and rotational motion stabilization using optical flow technique," *Consumer Electronics, IEEE Transactions on*, vol. 48, pp. 108–115, Feb 2002. pages 35

[127] S. Liu, L. Yuan, P. Tan, and J. Sun, "Steadyflow: Spatially smooth optical flow for video stabilization," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 4209–4216, 2014. pages 35

[128] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis. PhD thesis. MIT," 2009. pages 35

[129] M. Lefébure and L. D. Cohen, "Image registration, optical flow and local rigidity," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 2, pp. 131–147, 2001. pages 36, 39

[130] I. Reducindo, A. R. Mejia-Rodriguez, E. Arce-Santana, D. U. Campos-Delgado, and G. Rizzo, "Non-rigid registration based on local uncertainty quantification and fluid models for multiparametric mr images," 2013. pages 36

[131] G. Postelnicu, L. Zöllei, and B. Fischl, "Combined volumetric and surface registration," *IEEE Trans Med Imaging*, vol. 28, pp. 508–22, 04 2009. pages 36

[132] S. L. Keeling and W. Ring, "Medical image registration and interpolation by optical flow with maximal rigidity," *Journal of Mathematical Imaging and Vision*, vol. 23, no. 1, pp. 47–65, 2005. pages 36

[133] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pp. 1498–1505, 2010. pages 36

[134] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "DTAM: dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, pp. 2320–2327, 2011. pages 36

[135] S. Periaswamy and H. Farid, "Elastic registration in the presence of intensity variations," *Medical Imaging, IEEE Transactions on*, vol. 22, July 2003. pages 39

[136] J. R. Cooper and N. Ritter, "Optical flow for validating medical image tion.," in *The 9th IASTED International Conference on Signal and Image Processing*, pp. 502–506, IASTED/ACTA Press, 2003. pages 39

[137] P. Bertolino and A. Montanvert, "Multiresolution segmentation using the irregular pyramid.," in *ICIP (1)*, pp. 257–260, 1996. pages 40

[138] C. F. Neveu, C. R. Dyer, and R. T. Chin, "Two-dimensional object recognition using multiresolution models," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 1, pp. 52 – 65, 1986. pages 40

[139] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6314 of *Lecture Notes in Computer Science*, pp. 241–254, Springer Berlin Heidelberg, 2010. pages 40

[140] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. pages 41, 49, 57

[141] R. Szeliski, *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer-Verlag New York, Inc., 1st ed., 2010. pages 46, 47, 49, 51, 53, 99

[142] A. C. Bovik, *The Essential Guide to Image Processing*. Academic Press, 2009. pages 47

[143] D. Sun, S. R. M. J., and Black, "Secrets of optical flow estimation and their principles," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2432–2439, 2010. pages 54, 59, 109

[144] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek, "Nonlinear structure tensors," *Image Vision Comput.*, vol. 24, pp. 41–55, Jan. 2006. pages 63, 64

[145] L. Zhang, L. Zhang, and D. Zhang, "A multi-scale bilateral structure tensor based corner detector.," in *ACCV (2)* (H. Zha, R. ichiro Taniguchi, and S. J. Maybank, eds.), vol. 5995 of *Lecture Notes in Computer Science*, pp. 618–627, Springer, 2009. pages 63, 64

[146] W. Förstner and E. Gülch, "A Fast Operator for Detection and Precise Location of Distinct Points, Corners and Centres of Circular Features," 1987. pages 63

[147] W. Joachim, "Multiscale texture enhancement," in *Lecture Notes in Computer Science*, pp. 230–237, Springer, 1995. pages 63

[148] M. D. Budde and J. A. Frank, "Examining brain microstructure using structure tensor analysis of histological sections.," *NeuroImage*, vol. 63, no. 1, pp. 1–10, 2012. pages 63

[149] J. Bigun, G. Granlund, and J. Wiklund, "Multidimensional orientation estimation with applications to texture analysis and optical flow," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, pp. 775–790, Aug 1991. pages 63

[150] D. Butler, J. Wulff, G. Stanley, and M. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation," in *ECCV* (A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, eds.), vol. 7577, pp. 611–625, 2012. pages 67, 68, 102, 124, 125, 126, 129, 130, 133

[151] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. pages 69, 133

[152] M. Otte and H.-H. Nagel, "Optical flow estimation: Advances and comparisons," in *Proceedings of the Third European Conference on Computer Vision (Vol. 1)*, ECCV '94, (Secaucus, NJ, USA), pp. 51–60, Springer-Verlag New York, Inc., 1994. pages 69

[153] T. Lin and J. Barron, "Image reconstruction error for optical flow," in *In Vision Interface*, pp. 73–80, Scientific Publishing Co, 1994. pages 70

[154] R. Szeliski, "Prediction error as a quality metric for motion and stereo," in *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, (Washington, DC, USA), pp. 781–, IEEE Computer Society, 1999. pages 70

[155] "Front matter," in *Iterative Solution of Large Linear Systems* (D. M. Young, ed.), pp. iii –, Academic Press, 1971. pages 75, 99

[156] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust Statistics: The Approach Based on Influence Functions (Wiley Series in Probability and Statistics)*. New York: Wiley-Interscience, revised ed., Apr. 2005. pages 84

[157] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *Trans. Img. Proc.*, vol. 7, pp. 421–432, Mar. 1998. pages 84

[158] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers, "A convex formulation of continuous multi-label problems," in *Computer Vision – ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman, eds.), vol. 5304 of *Lecture Notes in Computer Science*, pp. 792–805, Springer Berlin Heidelberg, 2008. pages 89

[159] W. T. Freeman and E. H. Adelson, "The design and use of steerable filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 891–906, 1991. pages 90

[160] P. Gravel, G. Beaudoin, and J. A. D. Guise, "A method for modeling noise in medical images," *IEEE Transactions on Medical Imaging*, vol. 23, pp. 1221–1232, Oct 2004. pages 108

[161] H. Scharr, M. Black, and H. Haussecker, "Image statistics and anisotropic diffusion," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 840–847 vol.2, Oct 2003. pages 110

[162] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, Nov. 1992. pages 131

[163] D. Shulman and J.-Y. Herve, "Regularization of discontinuous flow fields," in *Visual Motion, 1989.,Proceedings. Workshop on*, pp. 81–86, Mar 1989. pages 133

[164] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, "Mav navigation through indoor corridors using optical flow," in *Proc. of The IEEE International Conference on Robotics and Automation (ICRA)*, May 2010. pages 150

Appendix A

Estimating Displacement Fields for Small Objects with Large Motion

Since the work of Horn-Schunck [1], many algorithms used the linearised data constancy assumption, this permitted researchers to convexify the optical flow problem. However the displacement field to be calculated should be small enough so that the linearisation holds. On the other hand some algorithms proposed the use of a non-linearised version of the data constancy assumption, such as the algorithm proposed by Brox et al. [2], although this also has its shortcomings. Since the data constancy is not linearised, the solution can be easily trapped in local minima, which was referred to as *'multi-modal'* in [2]. Therefore it would be useful to initialise the computation with a value close to the actual solution. The use of a Coarse-to-Fine framework (C2F) was adopted to solve this issue, where the computation of the flow field starts from a coarsened (down-sampled) version of the image. This solution is then propagated to a finer level, where it is used as an initialisation for the flow field at that resolution. This continues until the original resolution image is reached.

It is expected that the coarsened image will suffer from losing some details during the coarsening process. The flow field computed in the coarsened image will therefore be biased towards the larger structures remaining in that version of the image. Hence in the next finer level the initialisation is not close to the actual solution, but rather induced more by the larger structures appearing in the image. In addition, the calculation will fail if the small structures have displacements larger than its scale [13].

Brox et al. [13], [84] were perhaps among the first to note the problem with small structures and proposed to incorporate a descriptors' matching term into a variational framework. As descriptors have no problem in capturing displacements regardless of the magnitude of such

displacement, the descriptors term will steer the solution towards the correct displacements. The main problem with using descriptors is false matching, indeed the false matching may eventually deteriorate the overall optical flow computation. Other algorithms used patch matching to completely eliminate the use of C2F [53] , [89], or by exhaustive search for candidates [79]. The problem with such algorithms is the huge memory consumption and the slow speed of the matching process.

In this appendix, an extension to the steered-L^1 algorithm proposed in this thesis is going to be presented. The final aim of this research is to use the proposed algorithm in autonomous navigation and navigation assistant for the visually impaired. This is an initial stage of the research and contains only preliminary results. The specific objective of the algorithm presented in this appendix is to enable the steered-L^1 algorithm presented in this thesis (Section-4.3) to detect displacements of small objects in the scene, this is done by incorporating a descriptors matching term into the variational method discussed in Chapter-4. The incorporation of the descriptors' matching is inspired by the work of Brox et al. [13]. However an intrinsic difference is that the proposed algorithm here works in a primal-dual formulation and uses $TV-L^1$ which indicates a different optimisation method. The proposed algorithm here also has similarities with the method presented by Braux-Zin [98] which was also inspired by the work of Brox et al. [13], but the difference from the algorithm discussed in this appendix is the use of a different data terms.

A.1 Obstacle Avoidance, Corridor Traversing and Mobile Robot Navigation

Several algorithms proposed the use of optical flow as a cue to traverse corridors and avoid obstacles in mobile robots navigation [113], [115]. Some other examples also include Unmanned Aerial Vehicles (UAV) navigation, such as the Micro Aerial Vehicles (MAV) [164]. The use of optical flow for obstacle avoidance requires the algorithm to be able to estimate motion of small objects as well as larger objects. Consider the image sequence taken using a camera traversing a corridor. It is possible to compute the displacement field using one of the optical flow algorithms.

Fig. A.1 Corridor traversing.
Left: Image taken at time t. Right: Image taken at time t+1.
While the camera is moving forward, the ball is moving fast in the other direction.

Figure-A.1 contains relative movement between the camera and the surrounding environment. While the camera is moving forward with slow pace, the ball is moving in the other direction with high speed. The Horn-Schunck [1] method is used to find the displacement field for this image sequence. The visualised estimated field is depicted in Figure-A.2.

Fig. A.2 Optical flow of the corridor sequence Figure-A.1.
It can be seen that the displacements field is not accurately estimated using the Horn-Schunck method.

As the figure reveals, the estimated displacement field does not include the accurate displacement of the small ball apparent in the image.

A.2 Extending the steered-L^1 Algorithm

In this section, the formulation of the extended steered-L^1 algorithm is presented. As pointed out, the extension is inspired by the work of Brox et al. [13] which incorporated descriptors matching into the variational optical flow energy function. The descriptor used here is the Histogram of Oriented Gradient (HOG) [83]. The next subsection includes a discussion for HOG, followed by another subsection illustrating the algorithm.

A.2.1 Descriptors: Histogram of Oriented Gradients (HOG)

Brox et al.[13] used several types of descriptors including HOG in their work that proposed an algorithm to find optical flow for small scale objects that have large displacements. In their work they carefully designed the matching algorithm, as false matching can drastically affect the solution. In this section HOG descriptors are discussed, how they are calculated and what are the main parameters affecting their performance.

Before the computation of HOG, the process may involve gamma/colour normalisation. Images can be either grey-scale or represented using any colour space, e.g RGB or LAB colour spaces. HOG relies on computing histograms over a fixed window size, where each pixel contributes to the histogram. Image gradients are found using a certain kernel. Several kernels may be used, e.g. simple 1-D centred kernel $[-1,0,1]$, the cubic-centred $[1,-8,0,8,-1]$, 2-D ones like the 2×2 kernels, or even 3×3 like the Sobel kernel. The choice of the gradient kernel may affect the results as was shown in [83].

In the next step histograms are created for a small rectangular neighbourhood, this can have any size (e.g. 7×7). These rectangular neighbourhoods are called cells, cells can also be radial. Each pixel in these cells contributes to a histogram bin based on its gradient orientation, the contribution of the pixel is a function of the gradient's magnitude. The histogram is divided into equally spaced bins over the range of $0^o - 180^o$ in case unsigned gradients are considered. In the signed gradients case the range is $0^o - 360^o$. The cell histogram is not very discriminative by itself, especially at areas undergoing local illumination changes. For this reason, cells are grouped into blocks and the final descriptors are a collection of cells' histograms. Thus, each descriptor consists of $n\times m$ entries, where n is the number of cells in each block, and m is the number of bins of each histogram. To make the descriptors more robust to local illumination changes, contrast normalisation for histograms can be performed for each block (descriptor). Figure-A.3 depicts one descriptor.

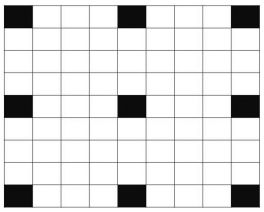


Fig. A.3 A single HOG descriptor.
This HOG descriptor consists of a collection of 9 histograms highlighted here in black [13].

Unlike SIFT features [85] which are computed at sparse locations, HOG descriptors can be computed in a dense grid. This means that a huge number of descriptors in each image can

be expected, and it will be computationally expensive if descriptors' correspondences are to be found. To reduce the computation complexity, it is possible to reduce the number of descriptors by picking only a number of these descriptors. This is done by considering only descriptors at equal spaces in the image grid. For instance by considering a descriptor every fourth pixel as in [13]. In addition to that the work in [13] included a further descriptors number reduction by ignoring descriptors in smooth image areas where no structures are available. This is done by calculating the eigenvalue for the structure tensor at similar locations to the descriptors. Descriptors with eigenvalues less than one eighth of the average of all the eigenvalues across the image are ignored.

The correspondence of these descriptors can be found as the descriptors with minimum distance between them, where the distance is the sum of squared differences between the descriptors (histograms). To speed up the matching process, specialised algorithms may be employed, such as the approximate nearest neighbour search. To further improve the matching process and in order to remove false matches, a backward consistency check was used in [13], where for each matched pair found from I_1 to I_2, we check if they still have the minimum distance between them if the calculation was done from I_2 to I_1, if not then these descriptors are ignored, and displacement at this pixel is set to 0.

A.3 Energy Function Formulation and Minimisation

In order to estimate an optical flow of a sequence of images and to improve the estimation accuracy of small objects in the scene, a descriptors matching term is added to Equation-4.24. Hence the energy function is written in the following way:

$$E = \int_{\Omega} \left(\alpha E_{data}(I_1, I_2) + E_{desc} + \frac{1}{2\theta}(\mathbf{u} - \mathbf{z})^2 + E_{smooth}(\mathbf{u}, \nabla \mathbf{u}, I_1) \right) \tag{A.1}$$

where E_{desc} is the descriptors' matching term which can be formulated in the following way [13]:

$$E_{desc} = \delta(\mathbf{x}) \Psi(|\mathbf{u} - \mathbf{u_d}|^2) \tag{A.2}$$

where $|\mathbf{u_d} - \mathbf{u}|^2$ is the descriptors matching term, $\mathbf{u_d}$ is the sparse displacements of descriptors at certain pixel locations[1]. The value of $\delta(\mathbf{x})$ is equal to 1 if there is a descriptors at the

[1] The following source code was used http://www.cs.berkeley.edu/~katef/LDOF.html

pixel **x** and 0 otherwise. The rest of the terms are defined in Section-4.3. The minimisation of this energy function is split into two steps, primal and dual steps. The minimisation of the dual step is similar to the minimisation is Section-4.3.1, where **z** is kept fixed to find the minimisation of **u**.

The primal step in the current case is written in the following way:

$$E_{Primal} = \int_{\Omega} \Big(\alpha \Psi\big(|I_2(\mathbf{x}+\mathbf{u}) - I_1(\mathbf{x})|^2\big) + \gamma \Psi\big(|\nabla I_2(\mathbf{x}+\mathbf{u}) - \nabla I_1(\mathbf{x})|^2\big) + \beta\, \delta(\mathbf{x}) \Psi(|\mathbf{u}-\mathbf{u_d}|^2) + \frac{1}{2\theta}(\mathbf{u}-\mathbf{z})^2 \Big)$$

The aim of this step is to minimise **u** while keeping **z** fixed. This minimisers of this equation is found by setting the derivatives with respect to zx, zy equal to 0. The derivation of this set of equation is similar to the derivation explained in Section-4.3.3, the only difference is the inclusion of the descriptors matching term in this case. Setting the derivative equal to 0 yield the following set of equations:

$$\begin{aligned} \big[\alpha\, \Psi'_1 . I_{2x}^2 + \gamma\, \Psi'_2 (I_{2xx}^2 + I_{2yx}^2) + \beta\, \delta(\mathbf{x})\, \Psi'_3 + \frac{1}{\theta}\big] zx \\ + \big[\alpha\, \Psi'_1 . I_{2x} I_{2y} + \gamma\, \Psi'_2 . I_{2xy} (I_{2xx} + I_{2yy})\big] zy \\ = -\big[\alpha\, \Psi'_1 r_{t0} I_{2x} + \gamma\, \Psi'_2 r_{tx0} I_{2xx} + \gamma\, \Psi'_2 r_{ty0} I_{2xy}\big] + \\ \beta\, \delta(\mathbf{x})\, \Psi'_3 u_d + \frac{u}{\theta}. \end{aligned} \tag{A.3}$$

$$\begin{aligned} \big[\alpha\, \Psi'_1 . I_{2x} I_{2y} + \gamma\, \Psi'_2 I_{2yx} (I_{2xx} + I_{2yx}) + \frac{1}{\theta}\big] zx \\ + \big[\alpha\, \Psi'_1 . I_{2y}^2 + \gamma\, \Psi'_2 . (I_{2xy}^2 + I_{2yy}^2) + \beta\, \delta(\mathbf{x})\, \Psi'_3\big] zy \\ = -\big[\alpha\, \Psi'_1 r_{t0} I_{2y} + \gamma\, \Psi'_2 r_{tx0} I_{2xy} + \gamma\, \Psi'_2 r_{ty0} I_{2yy}\big] + \\ \beta\, \delta(\mathbf{x})\, \Psi'_3 v_d + \frac{v}{\theta} \end{aligned} \tag{A.4}$$

where Ψ'_3 is the derivative of Ψ_3, and $\Psi_3 = \Psi(|\mathbf{u}-\mathbf{u_d}|^2)$.

A.4 Preliminary Experiments

The algorithm is implemented in MATLAB. Details of implementation are similar to implementation discussed in Section-5.2.1, with two main differences. The first is the use of the descriptors matching term. The second smoothing image sequence using a Gaussian filter during the coarsening process [67], Images were smoothed using a Gaussian kernel of size 5×5 and standard deviation of 1.67 this helps to produce smoother displacement fields. Figure-A.4 depicts two examples of image sequences extracted from a film of a moving ball in a corridor. In this film the camera is moving forward at a certain speed and a ball is moving in the other direction with a relatively higher speed. These image sequences were extracted from a film taken using Sony DSC-V1 camera. The film were taken at frame rate of 25 frame per second, each frame has a resolution of 480×640. Gray-scale images are used in these experiments.

Fig. A.4 A moving camera in a corridor.
Two image sequences. Top row: sequence-1. Bottom row: sequence-2. Left column: First image in the sequence (at time t). Right column: Second image in the sequence (at time t+1).

Figure-A.5 depicts the optical flow field for these image sequences using several algorithm.

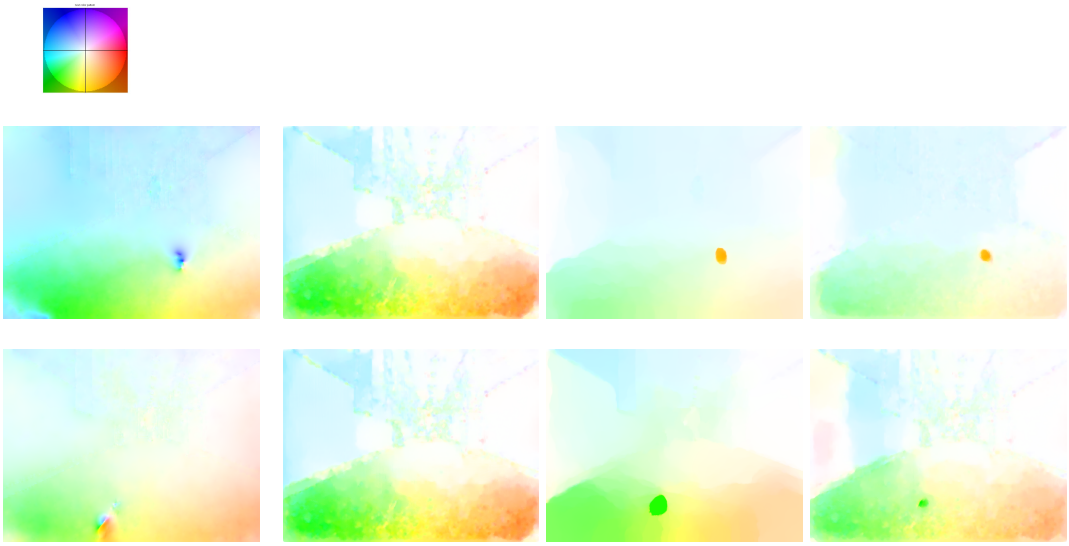


Fig. A.5 Results of corridor image sequences.
Top row: Image sequence-1. Bottom row: Image sequence-2.
First column(left): Optical flow field obtained via Horn-Schunck method. Second column: Optical flow obtained via steered-L^1 without descriptors matching $(\beta = 0)$. Third column: Optical flow obtained via LDOF [13]. Fourth column: Optical flow obtained via the method developed in this appendix with $(\beta = 7)$.

In Figure-A.5 the optical flow field obtained via the use of Horn-Schunck [1] failed to correctly estimate the displacement of the ball. The optical flow field obtained via the steered-L^1 algorithm presented in Chapter-4 also failed to detect this motion. On the other hand the optical flow field obtained via the algorithm discussed in this appendix is able to detect such motion.

A.5 Summary and Conclusion

In this appendix a primal-dual formulation for the algorithm proposed by Brox et al. [13] is presented. The aim is to enable the primal-dual optical flow algorithm discussed earlier in this thesis to detect the displacement fields of small objects in the scene. This method used the formulation discussed earlier in Chapter-4. Experiments were conducted using images produced specially for this purpose. The results demonstrate an improved ability to detect small objects that are visible in the scene. These results are still at an initial stage and further improvements on the algorithm are expected. The structure of primal-dual algorithms enables an easy implementation on modern graphical hardware [38] to obtain a

real-time performance. Real-time is very important for several applications such as robot navigation where the information is required to be processed in real-time.

Optical flow is an important cue in computer vision, and it has been used in many applications in mobile robot navigation in some old and other more recent research [113], [60], [115]. Navigation systems for the visually-impaired people is related to the area of mobile robot navigation. The goals in both cases can be similar, where it is required to traverse an area and avoid colliding with obstacles. As future work the algorithm presented in this appendix is to be part of obstacle avoidance system for the visually-impaired. This system is going to be inspired by early work on mobile robot navigation systems.

Appendix B

LOCAL-GLOBAL OPTICAL FLOW FOR IMAGE REGISTRATION

This paper appeared in iUBICOM '11: The 6th International Workshop on Ubiquitous and Collaborative Computing, 4 July 2011, Northumbria University, Newcastle

LOCAL-GLOBAL OPTICAL FLOW FOR IMAGE REGISTRATION

Ammar Zayouna Richard Comley Daming Shi
Middlesex University
School of Engineering and Information Sciences
Middlesex University, London NW4 4BT, UK
A.Zayouna@mdx.ac.uk R.Comley@mdx.ac.uk D.Shi@mdx.ac.uk

Registration is a fundamental task in image processing used to match two or more images taken, for example, at different times, from different sensors, or from different viewpoints. Optical flow is a technique in computer vision area to compute the displacement field of the contents within an image sequence. In the sense of correspondence, image registration and optical flow have very close relation. On the one hand, optical flow is used to do image registration; on the other hand, it is also used to evaluate the performance of image registration. In literature, either local optical flow or global optical flow is studied for image registration. In this paper, an improved optical flow technique, namely Local-Global, which combines the advantages of both techniques, is applied for image registration. Experiments are conducted to demonstrate the effectiveness of this method

Keywords: Image registration; global-local optical flow; correspondence estimation; tracking.

1. INTRODUCTION

As a fundamental task in image processing, image registration is defined as the process of aligning two or more images so that the shape, size, and spatial relationships of corresponding image contents can be easily matched or related. This concept is shown in Figure 1. The alignment process can be described with a geometrical transformation, namely a spatial mapping:

$$\mathbf{T}: f_B \mapsto f_A \Leftrightarrow \mathbf{T}(f_B) = f_A \qquad (1)$$

where $\mathbf{T}$ stands for the transformation, f_A is the target image, which acts as the reference and f_B is the source image, which is to be transformed toward to the reference.

The transformation can be found either by estimating the displacement of each pixel in the image or by finding certain number of parameters that describe the deformation pattern. The former is called nonparametric registration, while the latter is known as parametric registration.

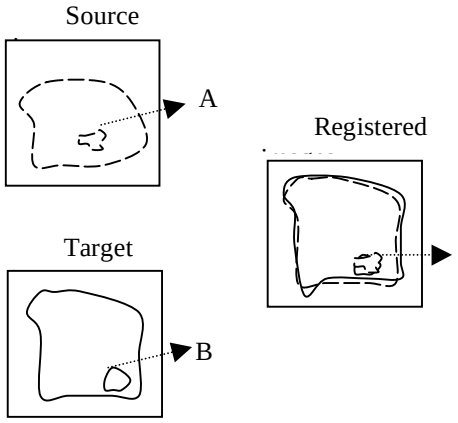


Figure 1: *Image registration: the correspondence between point A and point B is found after registration.*

Widely used in computer vision, optical flow is a technique to study the motion of contents within an image sequence. In literature there are two comprehensive papers on optical flow algorithms and their performance. (Barron et al., 1994) compared nine classic flow algorithms on the basis of accuracy and density. They classified these algorithms into four groups: differential techniques, energy based techniques, phase based techniques and region matching based techniques. (Liu et al., 1998) extended Barron's work by providing a coordinate system that compares accuracy with

efficiency. They classified optical flow algorithms into two groups: those that perform a gradient search on extracted structure of the image sequence and those that do not.

In optical flow, the motion is typically represented with velocity vectors originating from each pixel position. These vectors specify how the image pixels move between adjacent images. In this sense of correspondence, the procedure of determining optical flow is essential the nonparametric image registration. (Periaswamy and Farid, 2003) used optical flow in their general framework for medical image registration and achieved very good results. On the other hand, optical flow is also used to evaluate the result of medical image registration (Cooper and Ritter, 2003). A detailed description of the relation between image registration and optical flow can be found in (Lefébure and Cohen, 2001).

The most important assumption for optical flow is that when an image pattern moves, the brightness of a particular point in this pattern keeps constant. With this assumption, the famous optical flow equation can be derived either by chain rule differentiation or by first order Taylor approximation. For a 2D image, the problem is that the solution, namely the velocities in *x* and *y* directions (2 unknowns) cannot be uniquely determined with just one single equation. Extra constraints are needed. According to the types of constraints, differential techniques can be classified into local optical flow and global optical flow. Both of these two methods have their advantages and shortcomings. (Bruhn A and Weickert, 2005) proposed a method to combining the local optical flow to the global one.

However, when considered with image registration, all the work in literature only considers either of the optical techniques. In this paper, we aim to achieve improved image registration results by using the local-global optical techniques. The remaining part of this paper is as follows. Section 2 gives the details of local-global optical flow technique and the idea of apply it for image registration. Experiments are conducted in Section 3 to demonstrate the results and the quantitative registration errors are also given. Section 4 concludes this paper.

2. METHODS

In this section, we first discuss the original local and global methods, and further discuss how they can be combined and applied for image registration.

2.1 Local-global optical flow

According to experiments conducted, local method yields flow field which is more robust to noise and more accurate optical flow values. However, its operation time in MATLAB is much slower than global method. Another finding by the authors of this paper is that the larger number of iterations, the more accurate result will be. However, when the number of iterations reaches 500, the calculations of optic flow values will reach its limit.

Global method yields flow yields with 100% density, which means global method should fully represent the flow field, but are experimentally. Also, it is known to be more sensitive to noise. After we compile the global method program in MATLAB, the compilation time is less than one minute which is much faster than local method. However, Global method is not robust to noise.

Due to advantages and disadvantages of global and local methods, it is beneficial to combine these two methods to get a better one.

It is common to smooth the image values before calculating optic flow in order to get more clear results, and there are many existing smoothing techniques such as Gaussian filter using Gaussian calculations and Median filter which emphasize on the average values of each point. Combining the different smoothing effects can be done so that we can make use of the high robustness of local methods with full density of global techniques.

In an image domain Ω and time $t \quad [0, T]$ $g(x, y, t)$ represents the image sequence within the image. Gaussian filter has been chosen to make use of its low-pass effect in order to remove noise and other destabilizing high frequencies. However, it is noted that too much pre-smoothing should be avoided else it will remove the original features of the image.

$$f(x, y, t) := (K\sigma * g)(x, y, t) \tag{2}$$

where σ represents standard deviation. Normal flow (weighting factor) is given by

$$w_n = -\frac{f_t}{|\nabla f|}\frac{\nabla f}{|\nabla f|} \tag{3}$$

Let us look at how we local method is represented in (Bruhn A and Weickert, 2005).

$$E_{LK}(u,v) := K_p * ((f_x u + f_y v + f_t)^2)) \tag{4}$$

The standard deviation ρ of the Gaussian serves as an *integration scale* over which the main

contribution of the least square fit is computed. The larger the value of ρ, the more robust of the result will be under noise.

Global method is shown below in simplified form,

$$E_{HS}(u,v) = \int_{\Omega}((f_x u + f_y v + f_t)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2))dxdy \quad (5)$$

where Δ denotes the spatial Laplace operator $\Delta := \partial_{xx} + \partial_{yy}$

So, we can easily derive the final combined local-global method after some substitutions,

$$E_{CLG3}(\omega) = \int_{\Omega\times[0,T]}(\omega^T J_\rho(\nabla_3 f)\omega + \alpha|\nabla_3\omega|^2)dxdydt \quad (6)$$

and the weighted function has become

$$|\nabla_3\omega|^2 := |\nabla_3 u|^2 + |\nabla_3 v|^2$$

3. EXPERIMENTS AND RESULTS

In order to determine the result of Combined Local-Global optic flow method, we have implemented it in MATLAB, so as local and global methods. In the following session, we will compare and discuss the effects of different methods.

First, we chose two groups of photos taken from subsequent time points in two different videos; the camera speed is around 25 frames/second.

Figure 2: *Football match in time t1*

Figure 3:*.Football match in time t2*

Figures 2 and 3 are two photos taken with camera speed around 25 frames/sec. We used the two photos as inputs of global method, local method and combined local-global method in order to compare their results

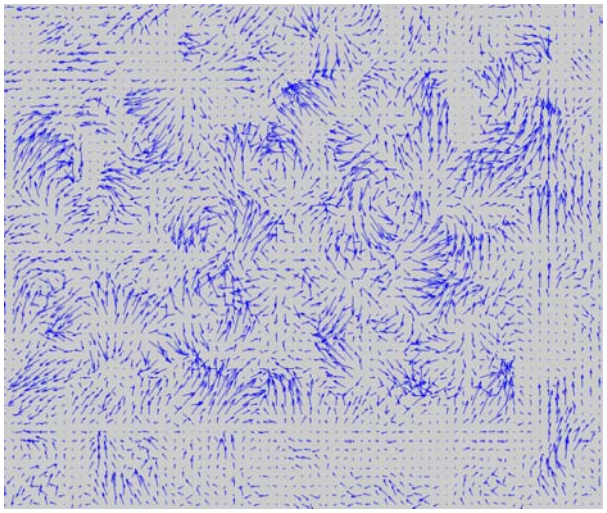


Figure 4: *Global football match*

In Figure 4, we can see that there is too much noise to affect the optic flow field, which means global method is not good enough to avoid the noises.

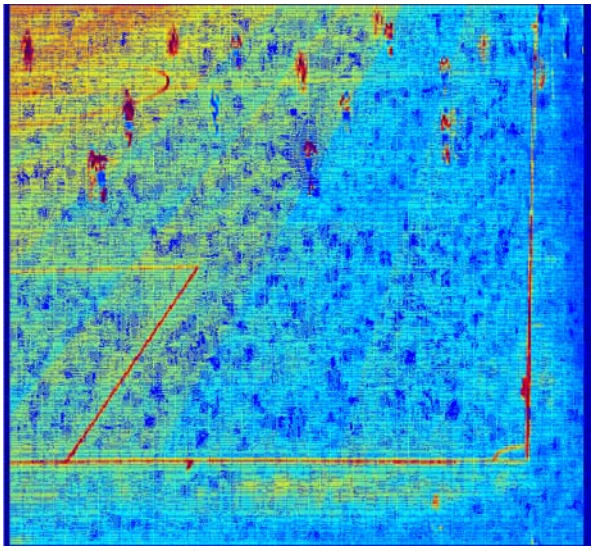


***Figure 5**: Local football match*

In Figure 5, the overall view is much better than that in global method because we can see clear flow field under noise interferences.

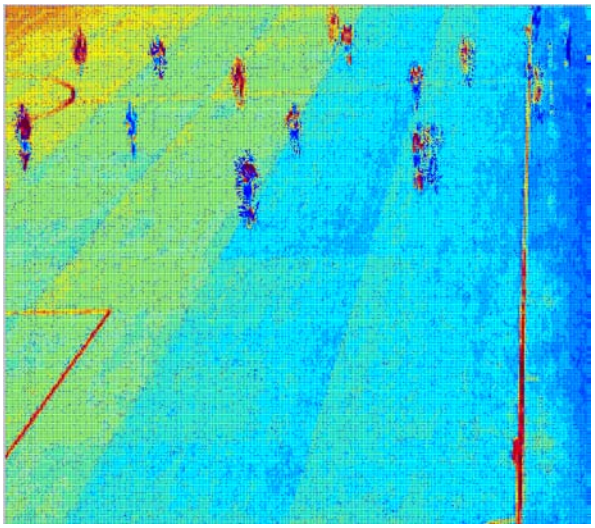


***Figure 6:** Combined local-global football match*

From Figures 4, 5 and 6 it is obvious that with the same inputs, the three different methods yield different output view. The three pictures are taken from the general view of the outputs, and it is clear that global method yields high density optic flow while local method yields more noise treatments. However, the CLG output show high contrast of optic flow density due to movement of objects to the stable objects and performs well under noise.

Generally speaking, from the overview pictures, optic flow in CLG output is more obvious around visible motion field while in local and global outputs; visible optic flow can be found throughout the whole pictures due to noise and other interferences. We chose the same part in Figures 4, 5 and 6 and zoom into the flow fields. The results are shown in Figures 7, 8 and 9.

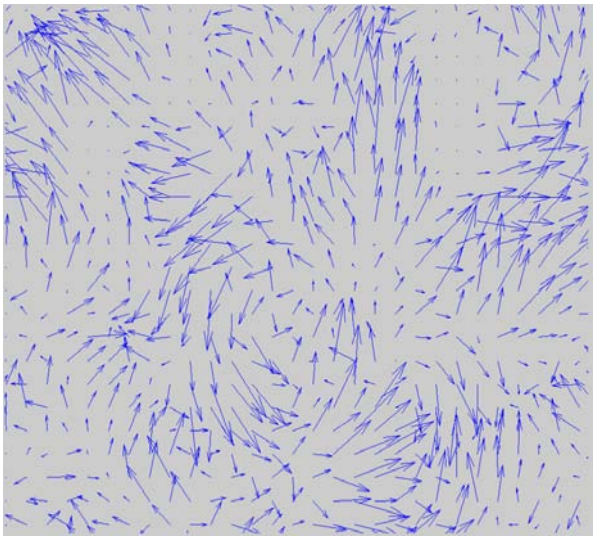


***Figure 7:** Zoom in Global football*

In Figure 7 above, global flow fields are smooth and with high density.

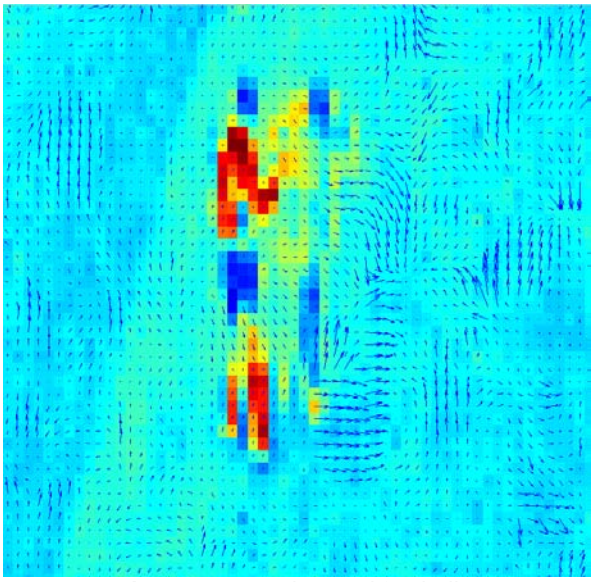


***Figure 8:** Zoom in Local football*

In Figure 8, local optic flow filed is more robust under noise and with clear difference between moving and non-moving objects.

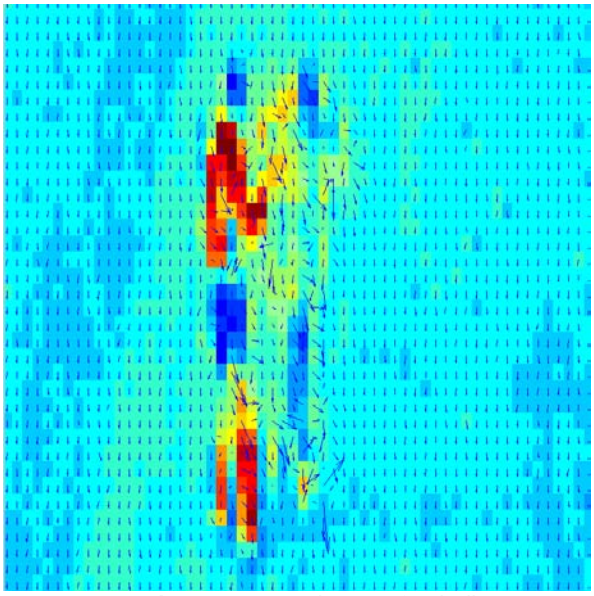


***Figure** 9: Zoom in CLG football*

In Figure 9, we can see that optic flow field in CLG is very distinguishing between obvious and unobvious movements and robust to noise which have better view than global and local methods.

4. CONCLUSION

In this paper, the relation between image registration and optical flow estimation is discussed. The advantages and shortcomings of both local and global optical flow are introduced. The improved optical flow technique, namely Local-Global, which combines the advantages of both techniques, is applied for image registration. The experiments on images with standard image sequences and images with synthetic deformations show that this method is highly effective with the advantages of both local and global optical flow techniques. Our future work will be focused on refine this method and applied on more images such as in medical area.

REFERENCES

BARRON, J. L., FLEET, D. J. & BEAUCHEMIN, S. S. 1994. Performance of optical flow techniques. Internation Journal of Computer Vision, 12, 43-77.

BRUHN A & WEICKERT, J. 2005. Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. International Journal of Computer Vision, 61, 211-231.

COOPER, J. R. & RITTER, N. J. Year. Optical flow for validating medical image registration. In: 9th IASTED International Conference on Signal and Image Processing, 2003. IASTED, 502-506.

LEFÉBURE, M. & COHEN, L. D. 2001. Image registration, optical flow and local Rigidity. Journal of mathematical imaging and vision, 14, 131-147.

LIU, H., HONG, T.-H., HERMAN, M. & CHELLAPPA, R. 1998. Accuracy vs efficiency trade-offs in optical flow algorithms. Computer Vision and Image Understanding, 72, 271-286.

PERIASWAMY, S. & FARID, H. 2003. Elastic registration in the presence of intensity variations. IEEE Transactions on Medical Imaging, 22, 865-874.