December 2015

Optical Flow At Occlusion Boundaries And In Occlusion Regions

Hua Meng
The University of Western Ontario

Supervisor
John Barron
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

© Hua Meng 2015

Follow this and additional works at: https://ir.lib.uwo.ca/etd
Part of the Other Computer Sciences Commons

Recommended Citation
https://ir.lib.uwo.ca/etd/3389

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact tadam@uwo.ca, wlswadmin@uwo.ca.
Abstract

Optical flow is an important research area in the Computer Vision field, with the estimation of optical flow at occlusion still an open problem. Traditional approaches to this problem have either used additional terms in a regularization calculation (the flow still tends to “bleed” across occlusion boundaries) or a local least squares calculation that attempted to minimize the influence of two adjacent differently moving regions on the optical flow at points close to both regions (the flow still tends to be “corrupted” by the two regions). Ideally, optical flow for two adjacent differently moving regions should be distinct right up to the occlusion boundary.

A recent approach to calculate optical flow at occlusion is to combining boundary and region segmentation with the optical flow computation. Based on the work of Sundberg et al. Arbelaez et al. and Brox et al., we implement a motion gradient ($mg$) edge map algorithm which detects motion information in closed regions in the image sequences. Here we utilize the motion gradient as an additional local cue in the globalized probability of a boundary ($gPb$) as a new boundary detector to produce a $gPb + mg$ contour map. The next step is to apply the Ultrametric Contour Map ($UCM$) mechanism, which is a framework to compute closed contours in a hierarchical region tree to produce a hierarchical edge map which indicates possible boundaries, including occlusion boundaries.

We implemented Sundberg et al.’s work to detect occlusion boundaries using optical flow, but, unlike Sundberg et al., we compute and display optical flow everywhere. The Sundberg et al. optical flow was generated by Brox et al’s method. They used a least squares calculation on the brox flow at pixels around an occlusion boundary to determine whether a boundary computed by the $gPb – UCM$ library developed by UC Berkeley is occluding or occluded. We extended their least squares idea to 1$^{st}$ and 2$^{nd}$ order optical flow models to generate dense optical flow inside each closed region. Finally, we analyze our optical flow fields both qualitatively and quantitatively. In particular, for quantitative analysis, we use warping error, as the correct flow is unknown. We show improved results over those of Sundberg et al., note a number of shortcomings in Sundberg et al.’s approach and point to areas of future research.

**Keywords:** Optical Flow, Occlusion Boundaries, Motion Gradient, UCM Contour Maps
Acknowledgements

I would like to thank my supervisor, Professor John Barron for the continuous support of my master thesis and related research, his patience, assistance and knowledge. He guides me from the beginning of writing of this thesis. I would like to thank my friends, mates, for their comments and encouragement. Last but not the least, I would like to thank all my family for supporting me spiritually throughout my study and life in general.
Contents

Abstract i

Acknowledgements ii

List of Figures vi

List of Tables xvii

1 Introduction 1

1.1 Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1

1.2 Discontinuous Optical Flow At Boundaries . . . . . . . . . . . . . . . . . . . 4

1.3 Thesis Contributions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

1.4 Thesis Outline . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2 Literature Survey 7

2.1 Classical Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

  2.1.1 Horn and Schunck Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . 9

  2.1.2 Lucas and Kanade Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . 10

2.2 Refined Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

  2.2.1 Yuan et al.’s Optical Flow . . . . . . . . . . . . . . . . . . . . . . . . . . 10
2.2.2 Niu et al.’s Optical Flow .............................. 11
2.2.3 Qualitative Comparison of Horn and Schunck’s, Lucas and Kanade, 
    Yuan et al. and Niu et al. Optical Flow ..................... 12
2.3 Brox et al. Optical Flow ................................. 13
2.4 Optical Flow and Occlusion .............................. 18
    2.4.1 Occlusion Detection ................................. 20
    2.4.2 Occlusion Filling ................................. 21
2.5 Stein and Herbert Occlusion Boundaries ................... 21
2.6 Sundberg et al.’s optical flow ........................... 22
2.7 Large Displacement Optical Flow .......................... 23

3 Motion Gradient ........................................ 24
    3.1 Motion Detector ..................................... 25
    3.2 Implementation of Motion Gradient .................... 27
    3.3 Motion Gradient Results ............................. 33

4 Optical Flow Re-estimation ............................... 39
    4.1 Globalized Probability of Boundary .................... 40
    4.2 The $gPb$ Detector ................................ 41
    4.3 Ultrametric Contour Map (UCM) ......................... 43
    4.4 $gPb+mg$ ........................................ 46
    4.5 Using the Boundary Floodfill Algorithm to Enumerate 
        Closed Regions ................................... 48
    4.6 Optical Flow Re-estimation ........................... 51
        4.6.1 Sundberg et al.’s Boundary Detector ............. 52
List of Figures

1.1 (a) The 10th frame of rocking-horse sequence from the rocking horse sequence [40] and (b) its optical flow field using Brox et al.’s optical flow algorithm [9] (vectors subsampled by 15 and scaled by 5.0) as implemented by Faisal and Barron [14]. 10 levels were used in the pyramid.

2.1 Motion flow Constraint Equation (from [7]).

2.2 Level 10 to level 2 Brox et al. optical flow using coarse to fine pyramidal structure [9] for the 10th frame of the rocking horse sequence.

2.3 The final optical flow field for the 10th frame of the rocking horse image sequence.

2.4 (a) and (b) are the 10th and 11st rocking horse images used for optical flow calculation while (c) shows the reconstructed 10th image using the 11st image and the optical flow at frame 10 (see Figure 2.3). Qualitatively, we can see the reconstructed and original rocking horse images look the same. Quantitatively, the warping error is 0.104764, which is quite small.

2.5 The optical flow for the 10th frame of the rocking horse image using Brox et al.’s method using 1 level.

3.1 \( Pb \) detector: Apply a circular window on pixel \((x, y)\) with radius \( r \). For 8 different angles denoted by \( \theta \), divide the circular window into two halves and compute the histogram difference between the two halves.

3.2 The 743th, 744th an 745th images of the tennis image sequence provided by University of California at Berkeley Computer Vision Group [18].

3.3 Derivative images \( D^- = I_t - I_{t-1} \) and \( D^+ = I_t - I_{t+1} \) for the tennis images.
3.4  The motion gradient images for the same radius and number of bins parameters
for both hard and soft histogram binning. ................................. 29

3.5  The top images are histograms of the two halves of the circular window for the
$D^-$ image at one pixel. The bottom images are histograms the two halves of the
circular window for the $D^+$ image at one pixel. ............................ 31

3.6  The 1st and 3rd columns are the $MG^-_r(x,y,\theta)$ and $MG^+_r(x,y,\theta)$ for the middle
tennis image. The 2nd column show the $MG_r(x,y,\theta) = \sqrt{MG^-_r(x,y,\theta) \cdot MG^+_r(x,y,\theta)}$
images for this image. The histograms used to compute the top motion gradient
images were created by “hard binning” while the histograms used to compute
the bottom images were created by “soft binning” (these results look better). 32

3.7  This figure shows how speed up the motion gradient calculation. We divide
the circular window into 16 slices. Before the calculation, we initialize the two
histogram distributions of the two halves using 8 slices separately. To update
the two histograms we subtract the first slice and add the next slice as shown in
the middle image in Green color slices. Doing this gives two new histograms
of the two halves of the circular window. ................................. 33

3.8  These motion gradient results are generated using radii of 3, 6 and 10, while
keeping the other parameters the same. Histogramming was done using soft
binning. The results show that bigger radii blur the motion gradient result. 34

3.9  These motion gradient results are generated using binning values of 3, 21 and
51, while keeping the other parameters the same. Histogramming was done
using soft binning. These results show that bigger binning values return more
responses. The results for binning value of 3 are not useful. 35
3.10 These results are screen captured from Sundberg et al. [44]. The top three frames are frames 743, 744 and 745 in the tennis sequence. The middle red images are the $MG^- (x, y) = \max_\theta [MG^-(x, y, \theta)]$ and $MG^+(x, y) = \max_\theta [MG^+(x, y, \theta)]$ from Sundberg et al. while the middle blue image is $MG(x, y) = \max_\theta [MG(x, y, \theta)]$, also from Sundberg et al. The benefits of combing $MG^-$ and $MG^+$ is to eliminate the double boundaries to obtain the correct motion gradient result is obvious. The bottom three images are our motion gradient results (provided here for comparison purposes), the bottom red images are $MG^- (x, y)$ and $MG^+(x, y)$ while the bottom blue image is $MG^+(x, y)$. The radius $r$ was 7 and binning value was 11 with soft binning being used.

3.11 The top three images (a)-(c) are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for the bench sequence using soft binning. The bottom three images (d)-(f) are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for the rocking horse images sequences, also using soft binning.

3.12 The top three images from left to right are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for (a)-(c) the hand sequence, (d)-(f) for the Hamburg taxi sequence and (g)-(i) for the trees sequence. Histogramming was done using soft binning.

4.1 (a) shows one of Berkeley Segmentation Dataset (BSDS300) benchmark original image, (b) shows the maximum response output by the $gPb$ boundary detector using UC Berkeley $gPb$ MatLab source code and (c) shows the UCM algorithm output using the $gPb$ result as input. Note that all regions are closed.

4.2 Results for the $gPb$ detector provided UC Berkeley Computer Vision Group’s MatLab source code [18]. The top images, (a) to (c), are original images. The bottom images are the maximal response result of $gPb$ detector over all orientations for (d) the 744th image of tennis sequence, (e) the 10th image of the rocking horse sequence and (f) the 10th image of bench sequence.
4.3 UCM results using the MatLab code provided UC Berkeley Computer Vision Group [18]. (a) 744\textsuperscript{th} frame of tennis sequence, (b) the 10\textsuperscript{th} image of the rocking horse sequence, (c) the 10\textsuperscript{th} image of the bench sequence, the UCM boundary map for the tennis image after thresholding (d) by 0.1 and (g) by 0.4, the UCM boundary map for the rocking horse image after thresholding (e) by 0.1 and (h) by 0.4 and the UCM boundary map for the bench image after thresholding (f) by 0.1 and (i) by 0.2.

4.4 The UCM boundary detection results produced from the 3\textsuperscript{rd} and 4\textsuperscript{th} bench with thresholding 0.2. We can see a spurious protrusion on the bench’s end and for the 4\textsuperscript{th} image and additional small closed region on the left.

4.5 These results are generated by the gPb-UCM algorithm and our mg algorithm. The top images are output of original algorithm, (a) is the gPb result for the 10\textsuperscript{th} frame of the rocking horse image sequence while (b) is the UCM result on the image in (a). The bottom images are output of gPb + mg-UCM algorithm, which is generated by combining our mg algorithm with gPb-UCM algorithm. The bottom left (c) is the gPb + mg result of the rocking horse image (we just added our mg result to the gPb) while (d) is the UCM result using gPb + mg as input.

4.6 These are results are generated using gPb + mg and UCM algorithms. The (a)-(c) images are gPb + mg results of the tennis (frame 744), rocking horse (frame 10) and the bench (frame 10) image sequences. The (d)-(f) images are the UCM outputs using gPb + mg (in (a)-(c)) as inputs. The (g)-(i) images are boundaries and closed regions results for these images after thresholding at 0.5 (tennis), 0.8 (rocking horse) and 0.2 (bench).

4.7 (a) and (b) are original Tai Chi diagram and rocking horse groundtruth images while (c) and (d) are the floodfilled output images. Each region is painted with a different color.
4.8 (a) and (b) images are boundary maps of the bench and tennis images while (c) and (d) are floodfilled region maps, again the regions are painted with different colors. ................................................................. 51

4.9 (a) is boundary image generated by thresholding the UCM output on $gPb+mg$, (b) motion feature image using $\delta$ Equation (4.14) and (c) $f$ image using Equation (4.15). ................................................................. 54

4.10 (a) is region map generated by our floodfill algorithm colored in Matlab jet(255) colormap using the occlusion boundary computed by $gPb+mg$ for the rocking horse. We color the boundaries as white while the different regions have different (non-white)colors. (b) This boundary detection result is generated by using Equations (4.16) and (4.17). we assign boundaries to the regions with the smaller distance and color it in that region’s color. .............................. 56

4.11 (a) is boundary image generated by computing the UCM output on $gPb+mg$, (b) the motion feature image using $\delta$ given in Equation (4.14) and (c) is the $f$ image using Equation 4.15. ................................................................. 56

4.12 (a) is region map generated by our floodfill algorithm colored in the Matlab jet(255) colormap using the boundary computed by $gPb+mg$. We color the boundaries white while coloring the regions non-white. (b) This result is generated using Equations (4.16) and (4.17). We assign boundaries to regions with the smaller distance and color it in that region’s color. .............................. 57

4.13 (a) is boundary image generated by computing the UCM output on $gPb+mg$, (b) the motion feature image using $\delta$ given in Equation (4.14) and (c) is the $f$ image using Equation (4.15). ................................................................. 57

4.14 (a) is region map generated by our floodfill algorithm colored in the Matlab jet(255) colormap using the boundary computed by $gPb+mg$. We color the boundaries white while coloring the regions non-white. (b) This result is generated using Equations (4.16) and (4.17). We assign boundaries to regions with the smaller distance and color it in that region’s color. .............................. 58
4.15 (a) is boundary image generated by computing the UCM output based on $gPb$, (b) the motion feature image using $\delta$ given by Equation (4.14) and (c) the $f$ image using Equation (4.15). .................................................. 58

4.16 (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by $gPb$. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color. .................................................. 59

4.17 (a) is boundary image generated by computing the UCM output based on $gPb$, (b) the motion feature image using $\delta$ given by Equation (4.14) and (c) the $f$ image using Equation (4.15). .................................................. 59

4.18 (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by $gPb$. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color. .................................................. 60

4.19 (a) is boundary image generated by computing the UCM output based on $gPb$, (b) the motion feature image using $\delta$ given by Equation (4.14) and (c) the $f$ image using Equation (4.15). .................................................. 60

4.20 (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by $gPb$. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color. .................................................. 61
4.21 (a) is the computed UCM boundary while (b) is the region map generated by our floodfill algorithm, colored using the Matlab jet(255) colormap using that boundary. We color the boundaries white while different regions are colored with different (non-white) colors.

4.22 (a) is the optical flow estimation using Brox et al. while (b) is new estimated optical flow using our 1st order method. We show the optical flow in every region in different colors. We can see that the affine model result is a smooth optical flow for each region.

4.23 (a) is new estimated optical flow using our 1st order method overlaid on the original image, with white pixels indicating the detected boundaries. (b) is new optical flow at the boundaries colored with the regions’ colors they belong to with white pixels indicating the detected boundaries. (c) and (d) show the same flow fields as in (a) and (b) but they are overlaid on the colored regions map.

4.24 (a) and (b) are new estimated optical flow using our 1st and 2nd order method while every region colored by region color so as to observe the performance inside regions.

4.25 (a) is new estimated optical flow using our 2nd order method overlaid on the original image with white pixels indicating the detected boundaries. (b) is new optical flow at the boundaries boundaries, colored by the regions’ colors that they belong to, again with white pixels indicating the detected boundary pixels. (c) and (d) are (a) and (b) but with optical flow overlaid on the colored regions map.

5.1 (a) is boundary image output by using the UCM algorithm for the tennis image and (b) is regions map generated by our floodfill algorithm.
5.2 (a) is colored Brox et al. optical flow for the tennis image. (b) and (c) is the 1\textsuperscript{st} order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 1\textsuperscript{st} order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white. ................................................................. 68

5.3 (a) is colored Brox et al. optical flow for the tennis image. (b) and (c) is the 2\textsuperscript{nd} order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 2\textsuperscript{nd} order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white. ................................................................. 69

5.4 (a) is boundary image output by using the UCM algorithm for the bench image and (b) is regions map generated by our floodfill algorithm. ............................................. 70

5.5 (a) is colored Brox et al. optical flow for the bench image. (b) and (c) is the 1\textsuperscript{st} order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 1\textsuperscript{st} order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white. ................................................................. 81

5.6 (a) is colored Brox et al. optical flow for the bench image. (b) and (c) is the 2\textsuperscript{nd} order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 2\textsuperscript{nd} order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white. ................................................................. 82

5.7 The left column images are warping error images of Brox et al.’s method, The middle column images are warping error images of 1\textsuperscript{st} order method, The middle column images are warping error images of 2\textsuperscript{nd} order method. They are computed on rocking horse, tennis and bench sequences. ........................................ 83
5.8 The left column reconstructed images are the rocking horse, (a) is the original image. (d) is the Brox et al. reconstructed image, (g) is the 1st order reconstructed image and (j) is the 2nd order reconstructed image. (b), (e), (h) and (k) are the respective images for the tennis image while (c), (f), (i) and (l) are the respective images for the bench image. 

5.9 (a), (d) and (g) shows the rocking horse images in rectangle [80 80 480 460], (b), (e) and (h) show the tennis image in rectangle [70 180 360 360] and (c), (f) and (i) show the bench images in rectangle [1 100 190 320]. The top images are for Brox et al., the middle images are for the 1st order method and the bottom images are the 2nd order model. The rectangles in red indicate the region of interest. Each image shows the RMS warping error for that image. 

5.10 We see some difference in our 1st and 2nd results of tennis image on the ground pats. Here we also show their difference in rectangle [250 1 360 360].

5.11 From top to bottom are the reconstructed images using Brox et al.’s method and our 1st and 2nd order method for each region’s warping errors displayed for the rocking horse image.

5.12 From top to bottom are the reconstructed images using Brox et al.’s method and our 1st and 2nd order method for each region’s warping errors displayed for the bench image.

A.1 These are results output by the UCM algorithm. The top images are original images. The upper middle images are the output of the gPb algorithm for 6th image of hand image sequence, the 10th image of the Hamburg taxi image sequence and 10th image of the trees image sequence. The lower middle images are the UCM results using gPb. The bottom images are the segmentation results for the UCM algorithm using gPb with thresholding at 0.8, 0.8 and 0.4 for the 3 image sequences.
A.2 Top images are the output for the $gPb + mg$ algorithm of 6th image of the hand images sequence, the 10th image of Hamburg taxi image sequence and of 10th image of the trees image sequence. The middle images are UCM results using $gPb + mg$ for the 3 images. The bottom image are the segmentation results of the UCM algorithm using $gPb + mg$ with thresholding at 0.9, 0.9 and 0.6 for the 3 images. ................................................................. 92

A.3 Sundberg et al.’s results for 6th image the hand sequence, including $\delta$, $f$, region map and distance assignment to regions. ................................................. 93

A.4 Sundberg et al.’s results for the 10th image of the Hamburg taxi sequence, including $\delta$, $f$, region map and distance assignment to regions. ...................... 94

A.5 Sundberg et al.’s results calculated on 10th image of the trees sequence, including $\delta$, $f$, region map and distance assignment to regions. ...................... 95

A.6 Top flows: 1st order optical flow for the 6th image of hand sequence, for colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. 97

A.7 Top flows: 2nd order optical flow for the 6th image of hand sequence, for colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. 98

A.8 Top flows: 1st order optical flow for the 10th image of Hamburg taxi sequence, for colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. ................................................................. 99

A.9 Top flows: 2nd order optical flow for the 10th image of Hamburg taxi sequence, for colored optical flow. Middle and lower flows: 2nd order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. ................................................................. 100
A.10 Top flows: 1\textsuperscript{st} order optical flow for the 10\textsuperscript{th} image of trees sequence, for both uncolored and colored optical flow. Middle and lower flows: 1\textsuperscript{st} order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. ................................................................. 101

A.11 Top flows: 2\textsuperscript{nd} order optical flow for the 10\textsuperscript{th} image of trees sequence, for both uncolored and colored optical flow. Middle and lower flows: 2\textsuperscript{nd} order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. ................................................................. 102

A.12 The warping errors calculated on hand sequence. The top images use Brox et al optical flow while the middle images use 1\textsuperscript{st} order flow and the bottom images 2\textsuperscript{nd} order flow. ................................................................. 103

A.13 The warping errors calculated on Hamburg taxi sequence. The top images use Brox et al optical flow while the middle images use 1\textsuperscript{st} order flow and the bottom images 2\textsuperscript{nd} order flow. ................................................................. 104

A.14 The warping errors calculated on trees sequence. The top images use Brox et al optical flow while the middle images use 1\textsuperscript{st} order flow and the bottom images 2\textsuperscript{nd} order flow. ................................................................. 105

A.15 1\textsuperscript{st} order method’s optical flow overlaid on the bench images from 006 to 013. . 107

A.16 1\textsuperscript{st} order method’s optical flow overlaid on the rocking horse images. . . . . . . 108

A.17 1\textsuperscript{st} order method’s optical flow overlaid on the tennis images from 740 to 747. . 109

A.18 2\textsuperscript{nd} order method’s optical flow overlaid on the bench images from 006 to 013. . 110

A.19 2\textsuperscript{nd} order method’s optical flow overlaid on the rocking horse images. . . . . . . 111

A.20 2\textsuperscript{nd} order method’s optical flow overlaid on the tennis images from 740 to 747. . 112
List of Tables

3.1 These data are computed on 2.4 Ghz Intel Core i7 processor and 8 GB 1600 MHz DDR 3 Memory. ......................................................... 33

5.1 We perform warping error calculation for: the 10th image of the rocking horse sequence and bench sequences and the 744th image of the tennis sequence for the Brox et al., 1st order and 2nd order flow fields. Zero flow means using vector (0,0) to compute warping errors as upper bound. Figure 5.7 shows the warping error images. .......................................................... 72

5.2 We perform Structural Similarity Index (SSIM) calculation for: the 10th image of the rocking horse sequence and bench sequences and the 744th image of the tennis sequence for the Brox et al., 1st order and 2nd order flow fields. .................. 73

5.3 We perform warping error calculation for 3 red rectangles in: the 10th image of the rocking horse sequence and bench sequences and the 744th image of the tennis sequence for the Brox et al., 1st order and 2nd order flow fields. Zero flow means we use vector (0,0) to compute warping errors as upper bound. .... 73

5.4 Region warping errors calculated by using Brox et al., 1st order and 2nd order methods for the 10th rocking horse image. ................................. 74

5.6 Region warping errors calculated by using Brox et al., 1st order and 2nd order methods for the 744th tennis image. ....................................... 79

5.5 Region warping errors calculated by using Brox et al., 1st order and 2nd order methods for the 10th bench image. ................................. 80
A.1 Warping error for the 6th image of hand sequence, the 10th image of the Hamburg taxi sequence and the 10th of trees sequence. ........................................... 96
A.2 Individual region warping errors calculated for Brox et al. and 1st and 2nd optical flow for 6th image of the hand sequence. ........................................... 96
A.3 Individual region warping errors calculated for Brox et al. and 1st and 2nd optical flow for 10th image of the Hamburg taxi sequence. ......................... 106
A.4 Individual region warping errors calculated for Brox et al. and 1st and 2nd optical flow for 10th image of the trees sequence. ................................. 106
Chapter 1

Introduction

In this thesis, we investigate some newest and best approaches for the calculation of boundary detection and optical flow fields, in particular for those boundaries that are occlusion boundaries. Occlusion occurs when one object with a distinct motion passes over another object with a different distinct motion. We would like to measure optical flow for these two objects, right up to the occlusion boundary that separates them. Towards our goal, we implement Sundberg et al.’s occlusion boundary algorithm [44] that uses optical flow to detect and label occlusion boundaries as occluding or occluded. This chapter explains some common concepts about optical flow, discontinuous optical flow at occlusion boundaries and why we need to do boundary detection before (or at the same time) computing optical flow.

1.1 Optical Flow

Optical flow is the apparent motion of image pixels from one image to the next in a sequence of images. 2D optical flow is the perspective projection of the relative 3D motion between an observer and the scene. We use the word “apparent” to denote our assumption that intensity changes between images is due entirely to the motion captures between the two images. Scene illumination changes are assumed not to happen or to happen so slowly that the effect is negligible over two frames. Scenes are assumed not to have uniformly textured objects (or
assumed to only have local uniformity) as such images can undergo motion but not exhibit any illumination changes.

We use vector fields to display this motion. Each vector indicate the direction and the speed of every pixel in the image. Some researchers use colour images because there is no need to subsample or scale the flow field. However, we find that subtle optical flow changes, such as those at an occlusion boundary, are not well displayed by such images.

Our datasets are from the University of California Berkeley Computer Vision Group Website ([18] and Stein and Hebert [40]), including the tennis, rocking horse, bench, hand, Hamburg Taxi (made by Nagel’s research group) and Trees (made by Shakey the Robot at Carnegie Mellon University) image sequences.

Many algorithms for estimating optical flow, from the classical methods proposed by Horn and Schunck [24] or Lucas and Kanade [30], both in 1981, to more modern approaches such as the hierarchical regularization method by Brox et al. [9] have been proposed. Consider the 10th frame of the rocking horse image sequence and its Brox et al. optical flow as shown in Figure 1.1. Our Brox et al. implementation was first implemented in C by Md. Faisal in 2007 [14]. He was able to obtain the same optical flow field as Brox et al. did for the Yosemite sequence. Later, this C code was converted to MatLab. As Figure 1.1 shows, we get some very good optical flow inside the horse’s head and at some (mostly right) parts of the background, but optical flow at occlusion boundaries and inside small occlusion regions is not very smooth.

A lot of optical flow work have been done in past, including two classical algorithms by Horn and Schunck (1981) [24] and Lucas and Kanade (1981) [30]. These algorithms have good overall performance. However, the optical flow quality is especially poor at occlusion boundaries. Optical flow at occlusion remains an open research area. In this thesis, we approach this problem by using not only an optical flow estimation method but also boundary detection. We first compute optical flow using the Brox et al. state-of-the-art algorithm [9], which is still considered one of the best algorithm for the estimation of optical flow. We often show the optical flow overlaid on the images so that the reader can see exactly how good optical flow is at various parts of the image. In particular, we overlay optical flow at occlusion boundaries
Figure 1.1: (a) The 10th frame of rocking-horse sequence from the rocking horse sequence [40] and (b) its optical flow field using Brox et al.’s optical flow algorithm [9] (vectors subsampled by 15 and scaled by 5.0) as implemented by Faisal and Barron [14]. 10 levels were used in the pyramid.

and display optical flow in each region using the regions’ colours.

In this thesis, occlusion boundary detection, is especially important. We wish to compute optical flow right up to occlusion boundaries, where the optical flow on both sides of the boundary are distinct and correspond to the unique image motions on each side of the boundary. We use Maire et al.’s work [31] and Arbelaez et al.’s work [3, 4] to do boundary detection. They developed a statistical probability map to indicate the probability of a pixel being a boundary pixel or not.

Our method for recalculating optical flow at occlusion boundaries is based on the Brox et al. optical flow algorithm we already have. We take Brox et al.’s flow as the original optical flow and then compute new optical flow based on it and boundaries produced by Maire et al.’s boundary detection algorithm. The reason we take boundary information into account is to improve the performance of optical flow at occlusion boundaries. We believe occlusion boundaries and their closed regions should have significant influence on the flow of all pixels inside such region. In particular, optical flow at an occlusion region should not “bleed” into adjacent occlusion regions. We do our experiments on three main image sequences:

1. the tennis court sequence from Sundberg et al [44],
2. the rocking horse and benchmark image sequences from Stein and Herbert [40] and

3. other image sequences (such as the Hamburg taxi sequence and the Shakey the robot sequence) that we have. These will be present in the appendices.

1.2 Discontinuous Optical Flow At Boundaries

The estimation of optical flow at occlusion is still an unsolved problem in Computer Vision. Optical flow at occlusion boundaries should exhibit discontinuities caused by moving objects or a foreground object moving against the background. Obviously, this happens a lot in natural images, where some objects moving differently pass over each other (occlude each other). Traditional optical flow algorithms do not work well in these areas. This is because when there are overlapping motions, many assumptions used by these techniques (for example, Horn and Schunck’s assumption that the flow field varies smoothly everywhere) are invalid. Zhang and Barron [49] investigated some newer optical flow methods specifically designed to work with occlusion. They found these algorithms indeed work better at occlusion boundaries but still exhibit many of the same kind of problems, but at a smaller scale. In this thesis, we follow their advice and do boundary detection first before estimating optical flow and which boundaries are occlusion boundaries. Sundberg et al.’s method [44] uses boundary detection and Brox et al. optical flow to determine which boundaries are occlusion boundaries. They never explicitly looked at the optical flow quality at and around these edges. The purpose of this thesis is to implement various algorithms extending from the idea proposed by Sundberg et al. [44] to improve optical flow at occlusion boundaries. We believe this will improve the performance of optical flow at boundaries. Also we will show results computed on different sequences, including tennis court, rocking horse, benchmark, trees and taxi sequences and reconstruct the first image from our optical flow and the second image to calculate quantitative evaluation numbers by comparing the first image and the reconstructed image (via a norm of the difference vectors).
1.3 Thesis Contributions

This thesis makes three contributions:

1. We have implemented Sundberg et al.’s occlusion detector motion gradient algorithm introduced in [44]. This gradient is used as another channel for boundary detector. We refer it as the motion cue. Traditional boundary detector typically doesn’t utilize motion information. In this case, having video or motion sequences, we can derive more information such as motion gradient used in boundary detection as another cue for the boundary detector.

2. Having these boundaries (closed occlusion contours), we try different methods to recalculate optical flow on boundaries and closed regions to produce a better results, including a first order derivatives, i.e. an affine model, described by Sundberg et al. [44] and a second order derivatives quadratic model (we added this model to the flow calculation). We are trying to use these closed boundary detection outputs to get better estimations of optical flow.

3. We compute a series of optical flow of image sequence to generate an optical flow video, which clearly show the object’s motion. Because we are focusing on the estimation of occlusion boundaries, we can think of various ways to estimate and describe the image motion at an occlusion boundary. Here we compute continuous optical flow sequences and put them together in order to generate a movie.

1.4 Thesis Outline

This thesis is organized as follows. Chapter 1 introduces the thesis. We introduce optical flow at occlusion. The contributions of the thesis plus an overview of its contents are given. We also explain various concepts, such as motion gradient, globalized probability of boundary, Orientated Watershed Transform and Ultrametric Contour Map. In Chapter 2, we present a general survey of some optical flow algorithms and boundary detectors. We utilize boundary
detector’s output to improve an optical flow calculation and we give a description of Brox et al.’s optical flow estimation method [9]. The Chapter 3 describes what a Motion Gradient is and how it is computed, how to use the Pb detector approach to calculate motion gradient (mg). We also show all results of our implementation and give the necessary details to explain how we implement this algorithm. In Chapter 4, we will introduce some other libraries we used in this thesis. Since we need to use gPb-UCM contour output as our input to generate gPb + mg UCM edge map, we give a description of it, introducing gPb and UCM. Also in Chapter 4, we introduce our own optical flow improvement algorithms; recalculated optical flow results of different image sequences are shown. Also we show the optical flow results using 1st and 2nd order models here. The remainder of experimental results are shown in Chapter 5. We evaluate our results quantitatively and qualitatively, comparing them to many other results. In Chapter 6 we give the conclusion and some future research idea
Chapter 2

Literature Survey

This chapter provides a general review of some work related to optical flow estimation and boundary detection. Optical flow is widely used in the Computer Vision field, for motion estimation and object detection, among other things. In this thesis, we will need to use a boundary detector in addition to computing optical flow. Thus, we also review some classic and common boundary detectors.

In Chapter 1 we discussed how discontinuities are common in natural image sequences and why this affects the accuracy of optical flow estimation. Many techniques have been proposed to estimate optical flow while preserving discontinuities in recent several years. We introduce some classical 2D optical flow methods in this chapter, including the algorithms by Horn and Schunck (1981) [24] and Lucas and Kanade (1981) [30] as well as other algorithms.

2.1 Classical Optical Flow

In Computer Vision, the estimation of optical flow is still a challenging problem, especially optical flow at occlusion boundaries. Barron et al. [6] and Fortun et al. Fortun-et-al-2015 have summarized various techniques for estimating optical flow. Most of these different optical flow algorithms are based on the motion constraint equation (sometimes also called the optical flow constraint line), which constrains the correct velocity at a pixel with spatio-temporal gradients
of $I_x$, $I_y$ and $I_t$ to fit a line defined by these parameters, as shown in Equation (2.1).

\[ I_x u + I_y v + I_t = 0 \]  

(2.1)

Figure 2.1: Motion flow Constraint Equation (from [7]).

In Figure 2.1, the normal velocity $v_\perp$ is defined as the vector perpendicular to the constraint line, which means is the velocity with the smallest magnitude on the motion constraint line. This equation is one linear equation in two unknowns, $u$ and $v$. That is to say, only the motion component in the direction of the local gradient of the image intensity may be estimated. This is called the aperture problem [7]. Here $\tilde{v}_\perp = v_\perp \hat{n}$ can be computed solely in terms of the intensity derivatives, $I_x$, $I_y$ and $I_t$ as:

\[ v_\perp = \frac{-I_t}{\|\nabla I\|_2^2} \quad \text{and} \quad \hat{n} = \frac{\nabla I}{\|\nabla I\|_2}, \]  

yielding:

\[ \tilde{v}_\perp = v_\perp \hat{n} = \frac{-I_t \nabla I}{\|\nabla I\|_2^2} \]  

(2.3)

$\tilde{v}_\perp$ is the raw normal velocity and $\nabla I = (I_x, I_y)$ is the spatial intensity gradient.

Pioneering optical flow algorithms were proposed by Horn and Schunck [24] and Lucas and Kanade [30]. We describe this work in more detail in the following subsections.
2.1.1 Horn and Schunck Optical Flow

Horn and Schunck [24] combine the motion constraint equation (in Equation 2.1) with a global smoothness term to constrain the estimated velocity field to vary smoothly everywhere:

$$||\nabla u||^2 + ||\nabla v||^2.$$ (2.4)

Horn and Schunck minimize:

$$\int_{D} \left( \nabla I^T \cdot \vec{v} + I_t \right)^2 + \alpha^2 \left( ||\nabla u||^2 + ||\nabla v||^2 \right) dx dy,$$ (2.5)

where domain $D$ means the whole image. The magnitude of $\alpha$ indicates the influence of the smoothness term (also known as the Lagrange multiplier) relative to the motion constraint equation. Typically, $\alpha$ can range from 1.0 to 10.0 to 100.0.

We can minimize the Equation (2.5) :

$$f(x, y, u, v, u_x, u_y, v_x, v_y) = (I_xu + I_yv + I_t)^2 + \alpha^2(u_x^2 + u_y^2 + v_x^2 + v_y^2)$$ (2.6)

to derive two iterative Gauss-Seidel equations, themselves in turn derived from the Euler-Lagrange equations of Equation (2.6):

$$u^{n+1} = \bar{u}^n - \frac{I_x \left[ I_x \bar{u} + I_y \bar{v} + I_t \right]}{(\alpha^2 + I_x^2 + I_y^2)}$$ and

$$v^{n+1} = \bar{v}^n - \frac{I_y \left[ I_x \bar{u} + I_y \bar{v} + I_t \right]}{(\alpha^2 + I_x^2 + I_y^2)}.$$ (2.7)

(2.8)

where $n$ denotes the iteration number, $u^0$ and $v^0$ denote initial velocity estimates, which are set to zero, and $\bar{u}^n$ and $\bar{v}^n$ denote $3 \times 3$ neighbourhood weighted averages of $u^n$ and $v^n$.

The problem with Horn and Schunck optical flow at occlusions can be attributed to the global smoothness constraint. Optical flow tends to “bleed” over occlusion boundaries. That is, the optical flow in one region slowly changes to the optical flow in a second adjacent region.
when the two regions have different motion and thus an occlusion edge exists between them. Such optical flow is very inaccurate.

### 2.1.2 Lucas and Kanade Optical Flow

Lucas and Kanade [30] assume constant velocity in local neighbourhoods and solve:

\[
\begin{bmatrix}
    u \\
    v
\end{bmatrix} =
\begin{bmatrix}
    \sum_i I_i^2 & \sum_i I_i I_{iy} \\
    \sum_i I_{ix} I_{iy} & \sum_i I_{iy}^2
\end{bmatrix}^{-1}
\begin{bmatrix}
    -\sum_i I_{ix} I_{it} \\
    -\sum_i I_{iy} I_{it}
\end{bmatrix}
\]

(2.9)

with the sums running from \( i = 1 \) to \( n \). Sometimes, the derivatives are weighed by values \( W_i \) that are inversely proportional to the distance of a pixel from the center pixel. A Gaussian function is typically used to compute such values. From empirical observations [6] it was found that Gaussian weights only improved the solution a negligible amount.

### 2.2 Refined Optical Flow

Some researchers have developed algorithms to estimate optical flow at occlusion. Most of these algorithms are based on Horn and Schunck and Lucas and Kanade classical methods we presented in the previous 2 subsections. For example, Yuan et al. [47] proposed a method based on Horn and Schunck’s model while Niu et al.’s [35] proposed a method based on Lucas and Kanade’s model. Both of these algorithms get better results over their original methods. However, they still have problems at occlusion, as we will discuss below. Zhang and Barron [49] implemented these two methods in 2D and 3D. Here, we focus on the 2D algorithms.

#### 2.2.1 Yuan et al.’s Optical Flow

Yuan et al. [47] investigated a modification of Horn and Schunck’s method with a variable weighting coefficient, a modified mean template based on intensity differences and a more efficient iterative method instead of Gauss-Seidel to solve the Euler-Lagrange equations. They
tested their modified algorithm on real image sequence and compare their results with Horn and Schunck’s results under the same conditions. Their results shows that the modified algorithm performs better than Horn and Schunck’s model in dealing with discontinuities at occlusion boundaries. The experimental results also showed that the compensating iterative algorithm is more efficient than the Gauss-Seidel iterative algorithm.

This new approach mainly focused on the adjustment of the weight coefficient, which modifies Horn and Schunck’s method and leads to a better preservation of flow discontinuities. Yuan et al.’s computation include:

1. Yuan et al. adopt variable smoothing weight coefficient instead of the constant coefficient $\alpha$ in the energy function.

2. a new mean template is used to calculate the average value of velocities which is more accurate compared to the symmetric template used by Horn and Schunck.

3. the Gauss-Seidel method is replaced by a new and more efficient iterative method to solve the Euler-Lagrange equations.

### 2.2.2 Niu et al.’s Optical Flow

Niu et al. [35] refined the flow estimate in a continuous region using the temporal neighborhood’s data while the flow estimate in a discontinuous region was computed using the local motion boundary information. They presented a dynamic over-constrained system of equations for dense, accurate flow computation to recover the optical flow and its deformation and acceleration. They compared their results with the results obtained by the optical flow fields obtained by Lucas Kanade [30] and others. The comparison demonstrated the expected accuracy of the proposed dynamic over-determined method by selective use of pixels in a point’s spatial and temporal neighborhood.

The Lucas and Kanade constant flow methods described above solves an overdetermined system by the least squares solution. However, that solution is robust to Gaussian noise but sensitive to outliers. In an optical flow computation, outliers can be caused by pixels on an
object boundary or an occlusion boundary. It is obvious that in such cases, the constant flow model is invalid. In Lucas and Kanade, the pixels within a neighborhood used to estimate optical flow are not selected, rather all pixels in the neighborhood are used.

Niu et al. [35] have devised a dynamic overdetermined optical flow solution with the goal of computing dense, accurate flow estimates with well preserved motion discontinuities. That is to say, their system avoids regularizing the irregular pixels, the outliers, and preserve discontinuities. To do so, they construct a dynamic overdetermined system of equations to recover optical flow and its deformation and acceleration. The main ideas of this technique can be described in the following steps:

1. Build up the initial equations at each pixel from selected pixels in its spatial neighborhood.
2. Test the obtained flow vectors using the brightness constraint equation.
3. Refine the flow vectors with selected temporal neighbors.
4. Detect possible occlusion or disocclusion regions with the help of acceleration vectors.

2.2.3 Qualitative Comparison of Horn and Schunck’s, Lucas and Kanade, Yuan et al. and Niu et al. Optical Flow

We ran Horn and Schunck, Lucas and Kanade, Yuan et al. and Niu et al. on the rocking horse image sequence. However, we do not show these flows because the image motion is too fast and the intensity derivatives are aliased. Poor flow fields result. Unfortunately we do not have access to hierarchical implementations of these methods. Brox et al. is a hierarchical method and with 10 levels it can measure this motion. For 1 level (no pyramid used) it cannot measure good flow, like these 4 methods. More details on Brox et al.’s performance for the rocking horse image is given in the next section.
2.3 Brox et al. Optical Flow

Many researchers, such as Sundberg et al. [44] (Brox is a co-author) believe that “variational optical flow is most accurate on edges”. A prominent variational optical flow method in that of Brox et al. [9]. Indeed this is the optical flow method used by Sundberg et al.

Brox et al. [9] proposed a hierarchical variational optical flow method. This method has produced the best optical flow for Yosemite Fly-Through sequence. The energy term of this variational method extends that of Horn and Schunck, by making constancy assumptions not only on brightness but also on the spatio-temporal gradient. Their method combines three assumptions: a brightness constancy assumption, a gradient constancy assumption and a spatio-temporal smoothness constraint, all using a coarse-to-fine warping strategy to measure larger optical vectors. Faisal and Barron [14] provided an implementation and evaluation of this algorithm in C. The assumptions:

1. The Brightness Constancy Assumption requires that the grayvalue of a pixel not be changed by the displacement of a pixel with velocity \((u, v)\) from time \(t\) to \(t + 1\):

\[
I(x, y, t) = I(x + u, y + v, t + 1).
\]  

(2.10)

This assumption uses the motion constraint equation \(I_x u + I_y v + I_t = 0\).

2. The Gradient Constancy Assumption requires that the spatial gradients of at a pixel be constant over time:

\[
\nabla I(x, y, t) = \nabla I(x + u, y + v, t + 1).
\]  

(2.11)

Here \(\nabla = (\partial_x, \partial_y)^T\) is for the spatial gradient.

3. The Smoothness Assumption assume a piecewise smooth flow field, which can be expressed as:

\[
|\nabla_3 u|^2 + |\nabla_3 v|^2,
\]  

(2.12)

where \(\nabla_3\) is spatio-temporal gradient \(\nabla_3 = (\partial_x, \partial_y, \partial_t)^T\). If only two images are available
for the optical flow calculation, \( \nabla_3 \) turns into the spatial gradient.

4. The Multiscale Approach requires an image pyramid be built for the 2 images one is computing optical flow for. The image reduction factor between levels in the pyramid is \( \eta \) (we use \( \text{eta} = 0.95 \) in this thesis). The idea is that image reduction over a number of pyramid levels slows down the image velocity magnitude. For example, with 10 levels and \( \text{eta} = 0.95 \), a 1 pixel/frame image velocity at level 1 becomes 0.6 pixels/frame at level 10. The coarse to fine strategy for computing fast optical flow (that would otherwise be aliased) requires a warping step at each level in the processing. The idea is to warp the computed optical flow out of the image at each level. Subsequent level optical flows only reflect the “corrected” optical flow needed to register the images. The sum of all these flows is the final flow.

Brox et al.’s energy model as proposed in [9] can be written as:

\[
E(w) = \int_\omega \Psi((I(x+w) - I(x))^2)dx + \gamma \int_\omega \Psi(\nabla I(x+w) - \nabla I(x))^2)dx + \alpha \int_\omega \Psi(|\nabla_3 u|^2 + |\nabla_3 v|^2)dx
\]

(2.13)

where \( \Psi(s^2) = \sqrt{s^2 + \epsilon^2} \) using \( L^1 \) functionals is a convex robust minimizer. \( \epsilon \) is a very small positive constant that prevents \( \Psi \) from being 0. Brox et al. choose \( \epsilon = 0.001 \). \( \alpha \) is a regularization parameter which determines the smoothness of the output.

We minimize \( E(w) \) by using the Euler-Lagrange equations and numerical approximation methods. They solve this nonlinear system of equations using SOR iterations. In order to estimate large displacement, they use the pyramid structure to downsample the images. The system of equations is solved in each level to get successive approximations of the optical flow.

Note that we still have some problems in the computation of optical flow at occlusion boundaries. From Stein and Herbert and Sundberg et al.’s work we can see that occlusion boundary detection is an important step in the estimation of boundaries’ optical flow.

In this thesis, we use a MatLab implementation of Faisal and Barron’s C code [14] of the Brox et al.’s method to compute Brox et al.’s variational optical flow. In this thesis, we build a
pyramid with 10 levels. At each level in the pyramid we save the full optical flow computed at that point. We end at level 1, with the final optical flow, \( u \) and \( v \).

Figure 2.2 shows the optical flow results at levels 10 to 2 while Figure 2.3 shows the final optical flow at level 1 for the 10\textsuperscript{th} frame of the rocking horse sequence.

Since the correct optical flow is unknown, direct quantitative analysis is not possible. To perform indirect quantitative analysis we compute warping error. Given two images, \( I_1 \) and \( I_2 \), and the optical flow, \( \mathbf{v} = (u, v) \), we use \( \mathbf{v} \) and \( I_2 \) to reconstruct \( I_1 \). The norm of the squared difference between the original \( I_1 \) and the reconstructed \( I_1 \) is due to either error in the computed optical flow and/or the error in the interpolation method used to warp \( I_2 \) back into \( I_1 \). At
pixel \((i, j)\) with floating point image velocity \((u, v)\) we need to compute the pixel intensity at \((i + u, j + v)\) and copy it to the reconstructed \(I_1\). To do this, we need to interpret the pixel values at the 4 integer positions surrounding this pixel at \((\lfloor(i + u), \lceil(j + v)\rfloor), (\lceil(i + u), \lfloor(j + v)\rfloor), (\lfloor(i + u), \lceil(j + v)\rceil)\) and \((\lceil(i + u), \lfloor(j + v)\rfloor)\). We use bilinear interpolation via MatLab’s \textbf{interp2} function. Figure 2.4.

![Final Brox Flow](image.png)

Figure 2.3: The final optical flow field for the 10\(^{th}\) frame of the rocking horse image sequence.

We also try to run Brox et al.’s method using 1 level shown in Figure 2.5. Aliased flow is true for other methods and that they need to be in a hierarchical framework like Brox to handle fast image motions.

The Brox et al. optical flow method returns a good estimation of optical flow. The major shortcoming of the algorithm is that it does not perform well at occlusion boundaries or at areas inside regions formed by closed occlusion boundaries. One idea we investigate in this thesis is to estimate optical flow independent inside each closed occlusion region. We need to do boundary detection and region detection to find all these locations before further processing. We use Brox et al optical flow, as suggested by Sundberg et al., to determine if each boundary
Figure 2.4: (a) and (b) are the $10^{th}$ and $11^{st}$ rocking horse images used for optical flow calculation while (c) shows the reconstructed $10^{th}$ image using the $11^{st}$ image and the optical flow at frame 10 (see Figure 2.3). Qualitatively, we can see the reconstructed and original rocking horse images look the same. Quantitatively, the warping error is 0.104764, which is quite small.

Figure 2.5: The optical flow for the $10^{th}$ frame of the rocking horse image using Brox et al.’s method using 1 level.

is an occlusion boundary. We give these details in later chapters.
2.4 Optical Flow and Occlusion

A complete (as of 2011) of the many approaches to computing optical flow at occlusion are given in an MSc thesis [48] and a conference paper [49]. We repeat here (next paragraph) verbatim the survey in the conference paper [49] and then add our contributions at the end.

Nagel [34] has proposed an “oriented smoothness” constraint that tries to suppress optical flow propagation in the direction orthogonal to the occluding boundaries in an attempt to stop velocity smoothing across motion boundaries. Bruhn et al. [11] presented a variational method to compute optical flow in real-time using bidirectional multi-grid strategies. Their energy functional regularizes a data term and an enhanced HS smoothness constraint that uses piecewise smoothness on both the image intensity distribution and the computed flow. Deriche et al. [13] imposes two conditions on the flow to allow discontinuities in the solution: isotropically smoothing the optical flow field inside homogeneous regions and preserving the flow discontinuities in the inhomogeneous regions by smoothing along curves with constant flow but not across them. Heinrich et al. [21] used a variational formulation where discontinuities are preserved by non-quadratic regularization using a modified $L^p$ norm. Schnörr [38] explicitly represented discontinuities by separating the stationary environment and moving object and then estimating flows over these separate domains. Heitz and Bouthemy [22] used a multi-modal, coarse-to-fine approach in a global Bayesian decision framework to estimate flow while preserving occlusion discontinuities. Guichard and Rudin [19] used the divergence of the flow and modelled occlusion using intensity-based matching and magnitude of divergence constraints. Ghosal and Vanček [17] used a ’weighted anisotropic’ smoothness term where locations with little available gradient information are more constrained and locations with strong intensity gradients (potential motion discontinuities) are less constrained. Proesmans et al. [36] computed flow using non-linear diffusion equations, in which information on optical flow bound-
Chapter 2. Literature Survey

aries is non-linearly fed back to the computed flow, preserving the discontinuities. [45] proposed a framework that simultaneously detects flow boundaries and uses those results in flow estimation. Flow analysis for basic Horn and Schunck, Horn and Schunck with boundary detection, edge-based flow estimation with boundary detection and edge-based flow estimation with boundary detection and a projection method were presented. Güler and Derin [20] detect discontinuities in noisy and textured images using weak continuity (using an elastic membrane energy model) and line configuration constraints in an graduated non-convexity optimization framework.

One item for future work in Zhang’s thesis [48] is to detect closed occlusion boundaries first, as described by Stein and Herbert [40] and Sundberg et al. [44] and then compute robust regularized flow within each closed occlusion boundary segment. Both Stein and Herbert and Sundberg et al. used optical flow in their occlusion boundary detection schemes but optical flow was only used to compute and label occlusion boundaries as either occluding or occluded. One interesting thing about Sundberg et al.’s work is that they detect closed occlusion boundaries (suggesting that optical flow could be computed inside such regions independently of optical flow elsewhere).

Note that occluded pixels have no correspondence in consecutive frames, and pixels in occluded regions may be overlaid by moving objects which can not be observed.

Optical flow estimation for images exhibiting both small and large displacements addressed by Brox and Malik [10] and Strecha et al. [41]. Because adjacent motions can differ so significantly, obviously there must be an occlusion boundary between them, although Brox and Malik do not explicitly compute such boundaries. Their algorithm combines correspondence with the Brox et al. [9] optical flow framework. Ayvaci et al. [5] consider occlusion detection as a variational optimization problem based on Lambertian reflection and static illumination assumptions. To detect moving objects or occlusion layers, their method optimizes the relaxed cost function using convex minimization. Bleyer et al. [8] combine image segmentation and graph-cuts optimization together to tackle the optical flow problem. Kolmogorov and Zabih
[28] utilize graph-cuts to handle occlusions, but they do it properly. They suggest that one pixel should correspond to at most one pixel in another image which is logical and reasonable. Heitz et al. [23] propose a multi-modal approach to solve the estimation of optical flow by using various complimentary constraints. They say multiple constraints will improve the accuracy of the estimation. Some work utilizes machine learning techniques to do boundary detection including Jacobson et al. [27] and Humayun et al. [25].

Fortun et al. [16] propose two steps to handle occlusion in the estimation of optical flow:

1. occlusion detection and

2. occlusion filling.

We address these two steps in the next two subsections.

### 2.4.1 Occlusion Detection

The occlusion detection process, for example, uses detectors to find boundaries and regions to segment the image into occluded and non-occluded regions. There is some work using this step to do optical flow estimation, such as Stein and Herbert [40], Sundberg et al. [44] or Ince and Konrad [26]. We discuss the work of Stein and Herbert [40], Sundberg et al. [44] in two subsections below.

First, these methods assume the existence of occlusions. Traditional methods for handing occlusion estimated optical flow ignoring occlusion, then identified occlusion boundaries and corrected optical flow in these areas. However, Ince and Konrad suggested interacting between optical flow and occlusion estimation to do this correction by using a variational formulation. They showed this extrapolation could lead to significant improvements in the estimation of optical flow over other approaches. Mozerov in [33] considered the estimation of optical flow as a matching problem and solved it by global optimization. Their approach to this goal used a pre-estimation of optical flow to correlate motion vectors frame by frame. Based on Brox et al.’s work [9], Li et al. [46] used an extended framework, refined based on coarse-to-fine strategy to recover correct details at various scales. Fortun et al. [15] suggested two steps to handle
large displacements. They supplied local motion candidates then combined them to estimate
global optical flow in a second step. They used a global regularized energy optimization to do
the aggregation step. Other work was based on a Bayesian framework, such as Strecha et al.
[41] and Smith et al. [39]. Also both Smith et al. [39] and Sun et al. [42, 43] utilized layered
model techniques.

2.4.2 Occlusion Filling

Fortun et al. distinguish between occlusion detection, i.e. segmenting the image into occluded
and non-occluded regions, and occlusion filling, i.e. recovering the missing flow subfields
in occluded regions. One suggestion to do this is to turn off the data term in the variational
calculation and just compute a smoother flow estimate for this region (since it is occluded
we cannot see it). In this thesis, we do not try to estimate the optical flow for occluded parts
of the image that cannot be seem. Rather, we are interested in computing optical flow in all
visible regions, especially right at moving occlusion boundaries, where potentially 2 surfaces
(the occluded and the occluding) have different motions.

2.5 Stein and Herbert Occlusion Boundaries

Stein and Herbert [40] initially over-segmentated the image and then with the help of a bound-
ary detector, identify occlusion boundaries. They provided a global reasoning model by learn-
ing the notion of fragment connectivity and constructing a factor graph to model fragment and
junction independences. They also provided some examples to compare the results of using
appearance cues only to those using both appearance and motion cues. Their results show
that appearance and motion classification performed together gave better results for fragment
detection and occlusion boundary detection. Stein and Herbert also provided a few occlusion
boundary detection examples that had consistently poor performance using their approach, due
to the extremely harsh lighting and the lack of texture. Finally, they proposed to extend their
approach to a per-frame basis over time as opposed to their current calculation for a single
reference frame in a short video clip.

### 2.6 Sundberg et al.’s optical flow

Sundberg et al. [44] propose a method of utilizing optical flow for occlusion boundary detection. Their work is based on the gPb-UCM technology and Brox et al.’s optical flow. They used a motion gradient algorithm to derive motion information from several frames to improve the boundary detector’s performance. Also they developed a strategy to decide which boundaries are occlusion boundaries using boundary detector’s output and optical flow data. In order to transform optical flow data into boundary information, they proposed a weighted least square approach to recalculate optical flow and then, by comparing the difference in the flow at each side of a boundary pixel to decide if that boundary pixel lies on the occlusion boundary.

Sundberg et al. report better occlusion boundary results than the boundary detector without motion information (in this case, we cannot distinguish between occlusion and non-occlusion boundaries). Inspired by their results, we decided to utilize their boundary detector output to improve the optical flow in the closed regions bounded by occlusion boundaries. We have implemented their motion gradient algorithm. We also utilize their gPb boundary detector and UCM to produce closed boundaries. Given such regions, we the recompute optical flow within each region using the original Brox et al. optical flow.

Originally, the idea was to detect closed occlusion boundaries first. Both Stein and Herbert [40] and Sundberg et al. [44] suggested to combine boundary detection and optical flow together to generate better results. Both of them used optical flow in their occlusion boundary detection schemes, but they did not show any optical flow results in their papers. Indeed, Sundberg et al. only compute optical flow for a small area around hypothesized occlusion boundaries. In the following chapters, we will introduce our implementation of the motion gradient and Sundberg et al.’s occlusion boundary detector.
2.7 Large Displacement Optical Flow

Brox and Malik proposed a method in [10] by combing various descriptors with the variational optical flow approach together to estimate optical flow. Also they developed a way to introduce features matching technique into optical flow field. The variation approach uses coarse-to-fine warping technique to compute the velocity, however, it is a kind of relaxing processing which will return incorrect optical flow at dense parts. They estimate large displacement optical flow using descriptor matching to improve the accuracy of dense optical flow. They use different terms such as color, gradient, matching and descriptors in their energy model. They perform region matching based on regions produced by segmentation method using SIFT and color descriptors. While these descriptors are computed in histograms of oriented gradients (HOG). Then they use continuation method to minimize the energy equation. The large displacement optical flow may be the next big challenge in the estimation of optical flow field.
Chapter 3

Motion Gradient

In the previous chapter, we presented many concepts about the estimation of optical flow. Discontinuous optical flow is a common phenomenon in natural images, these discontinuous may be caused by moving objects or partially occluded objects. Some optical flow methods focus on distinguishing occlusion boundaries from other boundaries. Stein and Herbert [40] and Sundberg et al. [44] try to detect the occlusion boundaries with the help of a boundary detector generated by combining motion cues, static cues and optical flow together. An evaluation of optical flow methods with energy terms to account for occlusions by Zhang and Barron [49] leads to the conclusion that detecting occlusion boundary can play an important role in estimation of the optical flow at occlusion boundaries.

Both Stein and Herbert and Sundberg et al. present algorithms for occlusion boundary detection that uses optical flow. Neither actually show the computed optical flows for their occlusion regions (regions with closed occlusion boundaries). Sundberg et al. use an additional motion cue, the motion gradient and with the information provided by the UCM boundary detector and the variational optical flow method by Brox et al. [9] they detected closed occlusion boundaries. They determined whether a boundary was an occlusion boundary by thresholding the difference between optical flow in the two adjacent regions: if the two optical flow were significantly different an occlusion boundary was hypothesized. Finally, they assigned figure/ground labels to these boundaries by comparing the optical flow on the boundary points
(the Brox et al. optical flow) with that of the regions adjacent to the boundary. In this way the region the boundary is moving with could be determined. Compared to Stein and Herbert, Sundberg et al.’s experimental result shows that their method is not only simpler but also better at detecting occlusion boundaries using the same datasets. We focus on the Sundberg et al. algorithm in this thesis.

Sundberg et al. [44] utilize the \( gPb \) algorithm (globalized Probability of boundary) introduced by Martin et al. [32] to do boundary detection. Then they use the OWT-UCM algorithm to generate a hierarchy of closed regions. They also use a new cue, named the motion cue, to perform a robust occlusion boundary detection. Based on the boundaries output by their detector and employing Brox et al.’s optical flow they design weighted least square equations to re-calculate optical flow on the boundaries. They didn’t show their optical flow results as their work is focused on the detection of occlusion boundaries.

In this chapter, we will explain our implementation of their motion gradient algorithm. Sundberg et al. introduce this idea in [44]. We will reproduce their motion gradient result before applying this algorithm on other image sequences that we use in our optical flow re-estimation algorithm.

### 3.1 Motion Detector

The boundary detector introduced by [32] produces \( Pb(x, y, \theta) \) which is the predicted posterior probability of a pixel being a boundary. It measures the differences of two halves of a disc of radius \( r \) at \((x, y)\) divided by a diameter at different angles \( \theta \). Martin et al. divide \( \theta \) into 8 orientations in the range \([0, \pi)\) for a particular radius.

Static cues like brightness, color, and texture are widely used in edge detection. These features are derived from static images. However, if we take image sequences into account, we can produce a motion cue as an new additional channel of information. This is the original idea why the use of the motion gradient is proposed by Sundberg et al. From their results, we can see that the motion cue provides a positive effect on the final occlusion boundary detection.
The motion gradient responds strongly to edges that are moving in the image. This allows to improve the quality and reliability of the occlusion boundary map.

In order to utilize the local motion cue information for the boundary detection we implement the computation of the motion gradients into 2 steps:

1. First, we compute the motion gradient, $mg(x, y, r, \theta)$, at every pixel in an image by applying $Pb$ detector methods.

2. Second, we compute the $\chi^2$ difference between two halves of the disc for all the orientations.

Figure 3.1 shows the gradient operator $G_r(x, y, \theta)$ used by Martin et al. [32] on every pixel in an image to produce the motion gradient values. At each pixel, we “draw” a disc centered at the pixel coordinate $(x, y)$ with radius $r$. Then we sample $\theta$ for 8 orientations in the range $[0, \pi)$ and divide this disc at these $\theta$ angles to compare the $\chi^2$ difference of the histograms of the two halves using a $\chi^2$ difference equation (below).

![Diagram](image)

Figure 3.1: $Pb$ detector: Apply a circular window on pixel $(x, y)$ with radius $r$. For 8 different angles denoted by $\theta$, divide the circular window into two halves and compute the histogram difference between the two halves.

This detector is widely used in boundary detection [32, 31, 3, 4]. The $Pb(x, y, \theta, r)$ operator consist of four parameters, including the pixel coordinates, $x$ and $y$, the orientation $\theta$ and the radius $r$. Often, researchers will use multiple scales, see, for example, Ren et al. [37], to
improve the performance of the boundary detector. However, Martin et al. [32] suggest that the best radius should have something to do with the image size while performing optimization (details are given in his paper). They have run their algorithm on hundreds images to find what scale is the best match. And they offer various weights for brightness, color, textures and gradient cues. We use their parameters directly in our work. The another question is how many orientations should we use. Martin et al. suggest for the purposes of optimization and quality, that 8 orientations seems to be enough. One can use more orientations, but it will not significantly improve the performance algorithm but will require more computational resources.

### 3.2 Implementation of Motion Gradient

We will implement this algorithm in Matlab. To illustrate our implementation, we use the tennis image sequence, provided by University of California at Berkeley Computer Vision Group [18] and shown in Figure 3.2. This is the same image sequence (actual frames are the same) used by Sundberg et al. [44].

![Frame 743, Frame 744, Frame 745](image)

Figure 3.2: The 743th, 744th an 745th images of the tennis image sequence provided by University of California at Berkeley Computer Vision Group [18].

Since color information is not needed for the motion gradient calculations, we transform the tennis images into grayvalue images before we use them. We normalize images so that all pixels are in the range [0 1]. The input of the motion gradient algorithm are these 3 frames, which are denoted as $I_{t-1}$, $I_t$ and $I_{t+1}$. 
We want to detect motions between each of the two frames, $I_{t-1}$ and $I_t$ and $I_t$ and $I_{t+1}$. We compute the difference images (as approximations to the 1st order intensity derivatives):

$$D^- = I_t - I_{t-1}$$

(3.1)

and

$$D^+ = I_t - I_{t+1}.$$ 

(3.2)

Since the three tennis images were normalized to be in range [0 1], $D^-$ and $D^+$ values are in the range [-1 1]. Figure 3.3 shows the derivatives (difference images), $D^-$ and $D^+$, computed for the three normalized tennis images.

![Figure 3.3: Derivative images $D^- = I_t - I_{t-1}$ and $D^+ = I_t - I_{t+1}$ for the tennis images.](image)

Before we apply the $Pb(x, y, \theta, r)$ detector to the two input images, we set up the boundary to be at least $r$ pixels away from the border. Because each pixel is the center of a circular window, to prevent boundary overflow we set the range of valid coordinates $(x, y)$ to be between:

$$r + 1 < x < MAX\_ROW - (r + 1)$$

(3.3)

and

$$r + 1 < y < MAX\_COL - (r + 1).$$

(3.4)
Because of these limits, the output boundary image may have an empty border with width $r$.

We apply the gradient operator $Pb(x, y, \theta)$ on every pixel in $D^-$ and $D^+$ to produce $MG^r_-(x, y, \theta)$ and $MG^r_+(x, y, \theta)$. That is, we draw a circular window on one pixel, then we sample $\theta$ for 8 orientations in the range $[0, \pi)$ and divide the window by a diameter oriented at $\theta$ degrees into two halves of a circular window. The half-disc regions’ pixels are described by histograms. There are different ways to create histogram distributions. In this thesis, we consider two methods: “hard binning” and “soft binning”. The number of binning values influence the output and we know of no optimal way to perform binning. Instead, we use a loop to try various number of bins to get the best result.

“Hard binning” means every pixel is assigned to one bin only while “soft binning” means assign potentially fractions of one pixel into a multiple adjacent bins based on an interval range by using linear interpolation. Martin et al. [32] claim that soft binning is often better than hard binning. We have implemented both binning methods, Figure 3.4 shows the results of hard binning and soft binning for the tennis with all other parameters keep the same. We can see that soft binning performs better than hard binning.

![hard binning](image1.png) ![soft binning](image2.png)

Figure 3.4: The motion gradient images for the same radius and number of bins parameters for both hard and soft histogram binning.

The process of applying the motion gradient operator requires one to process for each pixel the 8 orientations. For each pixel and each orientation for that pixel, we compute the
histogram of two image halves separately before a $\chi^2$ difference calculation. For each pixel, at each orientation, we will compute 2 histograms for the 3 image halves and compute their $\chi^2$ difference shown in Figure 3.5.

No matter which binning method we use (hard or soft), these histograms give pixel gray-value distribution information. The $\chi^2$ difference is used on the histogram distribution of the two halves to predict the posterior probability of a boundary at that image location for each orientations using:

$$\chi^2(g, h) = \frac{1}{2} \sum \frac{(g_i - h_i)^2}{g_i + h_i}$$  \hspace{1cm} (3.5)

As we said before, all pixels are normalized to be in the range [0 1]. If we distribute the normalized grayvalues into $n$ bins, the $\chi^2$ difference depends only on the count number and not the pixel grayvalues. Thus scaling the pixels (by normalization) will not effect the final $\chi^2$ difference values. According to Equation (3.5) the $\chi^2$ value may be greater than 1. In this case, we set all values $\geq 1$ equal to 1.

After we have computed all of a pixel’s gradient values for all orientations. We are only interested in the maximum responses of the motion gradients over all orientations for each pixel. We compute

$$MG^-_r(x, y) = \max_\theta \{MG^-_r(x, y, \theta)\}$$  \hspace{1cm} (3.6)

and

$$MG^+_r(x, y) = \max_\theta \{MG^+_r(x, y, \theta)\}$$  \hspace{1cm} (3.7)

respectively. Note that every movement will be detected at four locations, two in each of $MG^-$ and $MG^+$ separately. The one common location is the correct motion gradient. Sundberg et al. [44] suppress the spurious double responses in motion gradient responses by taking the geometric mean of $MG^-$ and $MG^+$ using:

$$MG_r(x, y, \theta) = \sqrt{MG^-_r(x, y, \theta) \cdot MG^+_r(x, y, \theta)}.$$  \hspace{1cm} (3.8)

We consider the maximal response output of this operator as the motion gradient $MG_r(x, y, \theta)$
value for pixel \((x, y)\) at angle \(\theta\) with radius \(r\). As we can see in Figure 3.6, from this we can compute the motion gradient \((mg)\) map corresponding to moving edges. We use this motion gradient map as an additional channel to produce a new global Probability of boundary \((gPb + mg)\). Figure 3.6 shows the results of our motion gradient algorithm for various radii using the two binning methods separately.

We can easily see that the most expensive part of the motion gradient computation is the the gradient calculation. At each pixel, we need to draw a circular window on it to compute the \(\chi^2\) difference at the various orientations. Martin et al. [32] suggest a way optimize the histogram by looping over all the orientations by dividing the circular window at \(n\) even orientations. We

![Figure 3.5: The top images are histograms of the two halves of the circular window for the \(D^-\) image at one pixel. The bottom images are histograms the two halves of the circular window for the \(D^+\) image at one pixel.](image)
Motion Gradient

Figure 3.6: The 1st and 3rd columns are the $MG_r(x,y,\theta)$ and $MG^+_{r}(x,y,\theta)$ for the middle tennis image. The 2nd column shows the $MG_r(x,y,\theta) = \sqrt{MG^-_{r}(x,y,\theta) \cdot MG^+_{r}(x,y,\theta)}$ images for this image. The histograms used to compute the top motion gradient images were created by “hard binning” while the histograms used to compute the bottom images were created by “soft binning” (these results look better).

divide the window into $2n$ slices. Then when we need to compute the histogram of two half discs, we sum the $n$ adjacent slices’ histograms to get two new half-disc histograms. A further speedup is possible. After we initialize a two half-disc histogram for one orientation, when we need to update this histogram we just subtract the first slice and add the next slice. This is shown in Figure 3.7 and denoted as “spin the disc” [32].

We have implemented the division of the circular window into $2n$ slices as the first step before we create any histograms for any half. During the loop calculation, we initialize the two halves of the histogram distribution from slice 0 to $n$ and from slice $n + 1$ to $2n$ by adding them together. After computing the $\chi^2$ difference, we just need to update the two halves’ histogram distribution by further subtracting and adding slices. For example, the first half needs to subtract slice 0 and add slice 9 while the second half needs to subtract slice 9 and add slice 0. This strategy works well, it greatly accelerates the computation process.

The execution time for rocking horse, tennis and bench sequence are listed in Table 3.2.
Figure 3.7: This figure shows how to speed up the motion gradient calculation. We divide the circular window into 16 slices. Before the calculation, we initialize the two histogram distributions of the two halves using 8 slices separately. To update the two histograms, we subtract the first slice and add the next slice as shown in the middle image in Green color slices. Doing this gives two new histograms of the two halves of the circular window.

<table>
<thead>
<tr>
<th>Image</th>
<th>Size</th>
<th>execution time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse 10th image</td>
<td>640x480</td>
<td>138.911</td>
</tr>
<tr>
<td>Tennis 744th image</td>
<td>530x380</td>
<td>99.3327</td>
</tr>
<tr>
<td>Bench 10th image</td>
<td>320x240</td>
<td>38.0785</td>
</tr>
</tbody>
</table>

Table 3.1: These data are computed on 2.4 Ghz Intel Core i7 processor and 8 GB 1600 MHz DDR 3 Memory.

### 3.3 Motion Gradient Results

When we experimented with the motion gradient algorithm, we found that two factors that affect the output the most were the radius and the binning values. But Martin et al. don’t say what the radius value they used is. We use a brute force, trial and error strategy by looking for the best result by looping over various radii and binning values. We use soft binning for our experiments reported in later chapters.

We could not find what the radius and binning values were for the motion gradient calculation in Sundberg et al.’s paper. We applied different binning values to compute the motion gradient to investigate the influence of radius. Figure 3.8 shows some motion gradient results for different radius values. Figure 3.9 shows some motion gradient results for different binning values. Our experiment results show that the bigger radius values blur the output while the
smaller radius values will make the output sharper.

We can compare the results in Sundberg et al. [44], as shown Figure 3.10. These images were screen captured from Sundberg et al.’s experimental images. Our results look “thicker”, while their result looks more cleaner and smoother. However, our images do capture the locations in the image where motion is occurring. Of course, even after extensive testing and debugging, we have done something wrong!!! Another possibility is that Sundberg et al. applied other preprocessing methods to the images (such as smoothing).

We also tested this algorithm on other image sequences, the rocking horse sequence as shown in Figure 3.11 and the trees image sequence as shown in Figure 3.12. We also have tested many different radii and binning values to get the best performance.

![Figure 3.8: These motion gradient results are generated using radii of 3, 6 and 10, while keeping the other parameters the same. Histogramming was done using soft binning. The results show that bigger radii blur the motion gradient result.](image)
Figure 3.9: These motion gradient results are generated using binning values of 3, 21 and 51, while keeping the other parameters the same. Histogramming was done using soft binning. These results show that bigger binning values return more responses. The results for binning value of 3 are not useful.
Figure 3.10: These results are screen captured from Sundberg et al. [44]. The top three frames are frames 743, 744 and 745 in the tennis sequence. The middle red images are the $MG^-(x, y) = \max_\theta\{MG^-(x, y, \theta)\}$ and $MG^+(x, y) = \max_\theta\{MG^+(x, y, \theta)\}$ from Sundberg et al. while the middle blue image is $MG(x, y) = \max_\theta\{MG(x, y, \theta)\}$, also from Sundberg et al. The benefits of combing $MG^-$ and $MG^+$ is to eliminate the double boundaries to obtain the correct motion gradient result is obvious. The bottom three images are our motion gradient results (provided here for comparison purposes), the bottom red images are $MG^-(x, y)$ and $MG^+(x, y)$ while the bottom blue image is $MG^+(x, y)$. The radius $r$ was 7 and binning value was 11 with soft binning being used.
Figure 3.11: The top three images (a)-(c) are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for the bench sequence using soft binning. The bottom three images (d)-(f) are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for the rocking horse images sequences, also using soft binning.
Figure 3.12: The top three images from left to right are the motion gradient images for $MG^-$, $MG$ and $MG^+$ for (a)-(c) the hand sequence, (d)-(f) for the Hamburg taxi sequence and (g)-(i) for the trees sequence. Histogramming was done using soft binning.
Chapter 4

Optical Flow Re-estimation

Our optical flow re-estimation starts with Sundberg et al.’s occlusion boundary and region detection. Edges are due to image brightness discontinuities, where the intensity changes sharply locally. While much edge detection work has been done it is still challenging to get accurate edge detection results. One unresolved problem with edge detection is how to know what a “correct” edge is: it is a highly suggestive judgment. Sundberg et al. propose one do contour or boundary detection. Contours/boundaries have some cognitive basis in a scene. It can be used to denote the boundary of an object, which means contour/boundary detection is something more than edge detection. Edge detection can give many edges, some of them may or may not be boundaries. Sundberg et al. actually take boundary detection one step further: they want to compute closed occlusion boundaries.

Various approaches have been introduced to perform edge detection. The most widely used edge detector is the Canny detector [12]. It performs its task using grayvalue discontinuities. This algorithm can be broke down into 4 main steps:

1. First, apply the Gaussian filter to attenuate noise,

2. Second, calculate the grayvalue intensity gradient of the image,

3. Third, apply non-maximum suppression to eliminate multiple responses to the edge detection and
4. Fourth, use two threshold values to choose edges.

The Canny detector is a simple detector producing reasonably good edge maps.

Other edge detector algorithms have been developed based on the Canny detector. Until recently, most of this work focused on reformulating the energy minimization term used by Canny. Now, researchers have started to utilize color and texture information in an image to derive better edges. Also some new methods are developed to calculate temporal gradients. The detector designed by Martin et al. [32] finds boundaries by using brightness, color and texture channels (information) in different orientations and scales. We have already discussed the motion gradient ($mg$) operator in the previous chapter. The call their boundary detector the $Pb(x, y)$ boundary detector. All information used by the $Pb(x, y)$ detector are derived from local cues. Martin et al. also use a global cue called spectral partitioning. Maire et al. [31] take both local and global cues into account to develop their globalized probability of boundary $gPb(x, y, \theta)$. This detector performs state-of-the-art boundary detection and is used by many other researchers [31, 44, 3, 4].

## 4.1 Globalized Probability of Boundary

In this thesis, we will use the MatLab source code for $gPb$ and UCM (Ultrametric Contour Map) detection, provided by University of California (UC) at Berkeley Computer Vision Group [18] to compute closed regions and boundaries [4]. Figure 4.1 shows their high quality boundary detection and segmentation result. The original image of this result is included in the source code package as an example.

We can see that both boundary detection and region detection performance is good. We will introduce $gPb$ and UCM detectors separately below to show their details.
Figure 4.1: (a) shows one of Berkeley Segmentation Dataset (BSDS300) benchmark original image, (b) shows the maximum response output by the $gPb$ boundary detector using UC Berkeley $gPb$ MatLab source code and (c) shows the UCM algorithm output using the $gPb$ result as input. Note that all regions are closed.

## 4.2 The $gPb$ Detector

The globalized probability of boundary ($gPb$) introduced by Maire et al. [31] and Arbelaez et al. [4] combines brightness, color and texture information together to do boundary detection in natural images. The $gPb$ detector is designed to detect boundaries accurately using multiple local and global cues. A global cue uses spectral partitioning. Like the earlier $Pb$ detector defined Martin et al. [32], the $gPb$ detector use the same gradient operators for brightness, color and texture channels, using them together to predict edge strength. It has been used in different tasks and has proven to be a high performance contour detector for transforming contours into regions [3] and localizing junctions [31], Ren [37] suggests using multi-scale to improve the quality of the result.

The $Pb$ detector defined by Martin et al. [32] was presented in the previous chapter and consists of the following steps:

1. place a circular disc at $(x, y)$ with different orientations $\theta$,

2. compute histograms of intensities in the two halves of the circular disc divided by a line
oriented at angle $\theta$,

3. compute the $\chi^2$ distance between the two histograms of the circular disc and

4. apply three scales of radii $r$ and eight values for $\theta$ (from $0^\circ$ to $180^\circ$) to calculate the maximum of all computed gradients.

They combine the multi-scale $Pb$ detector results, as suggested by [37], to get better performance, which can be referred to as $mPb$, for multi-scale $Pb$ detection. The $mPb$ detector is also used by Martin et al. [32] to include multiples scales for the computation of oriented gradients, such as brightness gradients (BG), color gradients (CGA/CGB) for LabA and LabB and texture gradients (TG) to improve quality. That is:

$$
mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta) (4.1)
$$

where $\alpha_{i,s}$ are weighting terms and $G_{i,\sigma(i,s)}$ are the different gradients in various scales $\sigma$, such as BG, CGA, CGB, TG.

As input to a spectral clustering stage, Arbelaez et al. [4] construct a sparse symmetric affinity matrix $W$ using the maximal value of $mPb$ along a line connecting two pixels. $mPB$ are multi scale cues [Arbelaez et al. use $\sigma/2$, $\sigma$ and $2\sigma$] for each of the brightness, color, and texture channels. Arbelaez et al. use $\sigma = 5$ for brightness and $\sigma = 10$ for color and texture. Then $W$ is created by connecting all pixels $i$ and $j$ within a fixed radius $r = 5$ with affinity:

$$W_{ij} = \exp(-p \in \tilde{ij} mPb(p)/\rho) (4.2)
$$

where $\tilde{ij}$ is the line segment connecting $i$ and $j$ and $\rho = 0.1$ is a constant.

They define the matrix $D_{ii} = \sum_j W_{ij}$ and solve for the generalized eigenvectors $v_0$, $v_1$, ..., $v_n$ and their eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_n$ of the system of equations $(D - W)v = \lambda Dv$. Treating each eigenvector $v_k$ as an image, they convolve it with Gaussian directional derivative filters at multiple orientations $\theta$ (the same as for the motion gradient calculation), obtaining oriented signals $\{\nabla_\theta v_k(x, y)\}$. The information from the different eigenvectors is then combined
to provide the “spectral” component of our boundary detector:

\[
s_{Pb}(x, y, \theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} \cdot \nabla_{\theta}v_k(x, y),
\]

where the weighting term \(\frac{1}{\sqrt{\lambda_k}}\) is motivated by the physical interpretation of the generalized eigenvalue problem as a mass-spring system.

The signals \(m_{Pb}\) and \(s_{Pb}\) convey different information, as the former is computed at all the edges while the latter extracts only the most salient curves in the image. Arbelaez et al. found that a weighted sum of the local and spectral signals was adequate to benefit from both behaviors:

\[
g_{Pb}(x, y, \theta) = \sum_s \sum_i \beta_{i,s}G_{i,\sigma(i,s)}(x, y, \theta) + \gamma \cdot s_{Pb}(x, y, \theta).
\]

\(g_{Pb}\) is rescaled using a sigmoid function (an example sigmoid function would be \(S(t) = \frac{1}{1+e^{-t}}\)) to allow a probabilistic interpolation. \(G_{i,\sigma(i,s)}\) are the different Gaussian gradients at various scales \(\sigma\) for the various cues BG, CGA, CGB and TG. Weights \(\beta_{i,s}\) and \(\gamma\) are determined using machine learning techniques. Full details for the computation of \(g_{Pb}\) are provided by Martin et al. [32], Maire et al. [31] and Arbelaez et al. [4]. The MatLab source code is available at the official UC Berkeley website [18].

We show \(g_{Pb}\) outputs for the tennis, rocking horse and bench images in Figure 4.2.

### 4.3 Ultrametric Contour Map (UCM)

In previous section, we gave a review of the boundary detection. However, we want to turn these boundaries into regions. The Orientated Watershed Transform and Ultrametric Contour Map, a state-of-the-art algorithms for contour detection and image segmentation was introduced by Arbelaez et al. [4, 3], offers a generic machinery to turn any contour detection results into a hierarchical region tree. They offer a method to segment of an image into closed regions automatically. They introduce a new algorithm using the Oriented Watershed Transform (OWT) to produce initial regions then use the Ultrametric Contour Map (UCM) [1] to generate
a hierarchy tree of regions. This algorithm offer a generic way to produce regions from the output of any edge detector. Full details are given in a number of papers [3, 1, 44, 4, 2].

Given the output of the $gPb$ detector, the $gPb$ oriented gradients values, we can transform them into an UCM to construct a hierarchy of regions. It will also provide the boundaries of segmentation at a given scale at threshold level $k$. Th UCM is represented as indexed hierarchy of regions or a soft boundary image. Figure 4.3 shows the results of UCM produced by the UC Berkeley Compute Vision Group MatLab source code [18]. We show UCM hierarchical map and thresholded image for each of 3 images.

Arbeláez et al. [4] presented a summary of the $gPb$-UCM technology and give a detailed description about how this detector works and how good its performance is. All techniques used in $gPb$-UCM are explained in this paper. They also compare their experimental results with other different algorithms to show how much better $gPb$-UCM performs over other algorithms. $gPb$-UCM provides universally better performance than alternative segmentation
Figure 4.3: UCM results using the MatLab code provided UC Berkeley Computer Vision Group [18]. (a) 744th frame of tennis sequence, (b) the 10th image of the rocking horse sequence, (c) the 10th image of the bench sequence, the UCM boundary map for the tennis image after thresholding (d) by 0.1 and (g) by 0.4, the UCM boundary map for the rocking horse image after thresholding (e) by 0.1 and (h) by 0.4 and the UCM boundary map for the bench image after thresholding (f) by 0.1 and (i) by 0.2.

algorithms [4].

The boundary produced by the UCM algorithm should be a closed boundary (regions adjacent to the image border include that part of the border as part of their closed boundary). Different threshold level $k$ may produce better results. We add our Motion Gradient into $gPb$ detector to produce our own $gPb + mg$ output (Sundberg et al. did this first).

UCM doesn’t always return accurate boundary detection results. This may be due to image quality, size or other reasons. Even different parameters will affect the final output. Figure 4.4
shows a lower quality UCM boundary detection result.

![3rd UCM bench contour](image1)
![4th UCM bench contour](image2)

**Figure 4.4:** The UCM boundary detection results produced from the 3rd and 4th bench with thresholding 0.2. We can see a spurious protrusion on the bench’s end and for the 4th image and additional small closed region on the left.

### 4.4 \textit{gPb} + \textit{mg}

The \textit{gPb}-UCM technology is a state-of-the-art boundary detector. Sundberg et al. [44] suggest an improvement in performance is possible by using the motion gradient (\textit{mg}) detector. In this thesis, given the \textit{gPb} detector algorithm, we add the motion gradient (\textit{mg}) detector to it using the same brightness, color and texture cues that Sundberg et al. did. Our way of combining the \textit{gPb} and the \textit{mg} detectors is to compute \textit{mPb} and \textit{sPb} normally but for the \textit{gPb} detector, we use brightness, color, texture, \textit{sPb} and \textit{mg}. Maire et al. [31, 4] learned the weight values for cues using machine learning techniques. In this work, we use 1.0 for the \textit{mg} weight value (this assumes all \textit{mg} values are correct). The \textit{gPb} + \textit{mg} oriented boundary map is produced by simply adding these weighted cues together, which then used UCM as input to produce closed regions and boundaries. We can write the cue summation as:

\[
gPbmg(x, y, \theta) = \sum_x \sum_y gPb(x, y, \theta) + mg(x, y, \theta)
\]  

(4.5)
Figure 4.5 images are generated using the UC Berkeley Computer Vision Group’s MatLab source code to perform boundary detection of the 10th rocking horse image gPb and UCM result. Next we compute the output of the gPb + mg detector using our implementation of mg and the UC Berkeley Computer Vision Group’s gPb algorithm. Lastly we show the boundary detection using UCM with gPb+mg as its input. These results show that our mg detector is computed correctly. We attribute the small differences to our mg detector results.

Figure 4.5: These results are generated by the gPb-UCM algorithm and our mg algorithm. The top images are output of original algorithm, (a) is the gPb result for the 10th frame of the rocking horse image sequence while (b) is the UCM result on the image in (a). The bottom images are output of gPb + mg-UCM algorithm, which is generated by combining our mg algorithm with gPb-UCM algorithm. The bottom left (c) is the gPb + mg result of the rocking horse image (we just added our mg result to the gPb) while (d) is the UCM result using gPb+mg as input.
It is clear that there is too much noise present after we add $mg$ to our detector without any preprocessing. The $mg$ detector is designed to detect motion information between adjacent images in a sequence, but some of the detection may result from boundaries or edges even that we don’t want. Normally, each cue is multiplied with a weight value which is learned from machine learning techniques to eliminate these fake responses. As we don’t use machine learning things in this thesis, we just add $mg$ directly using weight of 1.0. This means UCM can give over-segmentation results. Here we try different threshold values to find the best boundary and region results. Figure 4.6 shows $gPb+mg$, $gPb+mg$-UCM results and thresholded versions of these results for 3 different image sequences.

We haven’t found an effective way to find the optimal threshold values. The output of the UCM algorithm lay in the range [0 1], so we are able to try every possible threshold value in this range to threshold UCM results to obtain the best results. We can also designate a fixed number of regions to be. We can iteratively update the threshold value using the flood fill algorithm until the output regions number is smaller or equal than the maximum regions numbers. We introduce how we find regions using the floodfill algorithm in the next section.

4.5 Using the Boundary Floodfill Algorithm to Enumerate Closed Regions

The thresholded UCM results indicate which image pixels belong to boundaries and which image pixels belong to the background. In order to assign each region a region number we have implemented the floodfill algorithm to enumerate the regions. The floodfill\footnote{See any undergraduate Computer Graphics textbook for recursive and iterative algorithms for doing this.} algorithm fills different connected areas using different indices (the region numbers). The idea is start with assigning a arbitrarily chosen (say the upper left corner non-boundary pixel) single pixel a region number, then recursively check all adjacent pixels around that pixel to find out if those pixels belong to the same region. Each non-boundary pixel is label with a region number. If a pixel is unvisited, the algorithm will label it with the next available region number, then
Figure 4.6: These are results are generated using $gPb + mg$ and UCM algorithms. The (a)-(c) images are $gPb + mg$ results of the tennis (frame 744), rocking horse (frame 10) and the bench (frame 10) image sequences. The (d)-(f) images are the UCM outputs using $gPb + mg$ (in (a)-(c)) as inputs. The (g)-(i) images are boundaries and closed regions results for these images after thresholding at 0.5 (tennis), 0.8 (rocking horse) and 0.2 (bench).

flood fill that region, labelling all pixels in that region with the same region number. Figure 4.7 shows the floodfill algorithm results for two example groundtruth boundary images, the Tai Chi Diagram and the 10th frame of the rocking horse image sequence. Figure 4.8 shows the floodfill algorithm results for the computed boundary images for the bench and tennis images. In Figure 4.8b and 4.8d we can see that when too many regions are detected, it is difficult to label and color all regions distinctly. Usually too many regions means many small regions.

The floodfill result is our region map, in that it indicates which region each pixel belong to.
Figure 4.7: (a) and (b) are original Tai Chi diagram and rocking horse groundtruth images while (c) and (d) are the floodfilled output images. Each region is painted with a different color.

This region map not only include region information but also the boundary information about which pixels are part of a region. Using this region information we can re-estimate optical flow independently of pixels in other regions (which might be moving differently) for each region. This improves optical flow performance because we do not compute flow across occlusion boundaries. The following processes are dependent on region numbering, which makes this step very important. We have two possible regions maps, generated by $gPb$ and $gPb + mg$ separately with UCM. Sundberg et al. suggest the $gPb + mg$ boundary map is best because it is hypothesized that occlusion boundaries with different motion on each side are moving and non-occlusion boundaries are stationary.
Figure 4.8: (a) and (b) images are boundary maps of the bench and tennis images while (c) and (d) are floodfilled region maps, again the regions are painted with different colors.

4.6 Optical Flow Re-estimation

In this section we present Sundberg et al.’s optical flow calculation. This calculation is very computational expensive is is only perform in the neighbourhoods of hypothesized occlusion boundaries. Again, we emphasize that Sundberg et al. do not report any optical flow results. Rather they use the optical flow at hypothesized occlusion boundaries to determine if a boundary is an occlusion boundary and if s, whether it is an occluding or occluded boundary.

Given the region map for an image plus its Brox et al. global optical flow, we impose 1\textsuperscript{st} order (affine) and 2\textsuperscript{nd} order models on this flow in each region and re-estimate the optical flow.
4.6.1 Sundberg et al.’s Boundary Detector

Before showing our re-estimated optical flow, we review how Sundberg et al. do occlusion boundary detection by using boundary detection [the gPb + mg and UCM algorithms] and optical flow. They estimate the motions differences by looking at neighboring regions adjacent to every boundary pixel which is indicated by thresholded UCM output. They designed a weighted filter to ensure the estimation is not polluted by the motion of the edge itself or the motion of the adjacent regions. This filter computes a weight at pixel \((x, y)\) for each region that is used in the optical flow calculation at that point for that region. This weight is computed as:

\[
\begin{align*}
    w^R_i(x, y) &= \frac{1}{Z} \exp\left(\frac{(x-x_i)^2 + (y-y_i)^2}{2\sigma^2}\right) \delta(r(x, y), R) i_d(x, y) \\
    \end{align*}
\]

(4.6)

for each edge point \(P_i = (x_i, y_i)\) and neighboring region \(R\). We use \(Z\) as a constant (but if we knew it by other means we could use this value). Here we set \(Z = 1\). We use \(\sigma = 3.5\) and the radius of the window \(r = 7\) pixels as Sundberg et al. did. \(\delta\) in the equation is the Kronecker delta function (it is 1.0 when the argument is 0.0 and 0.0 otherwise), \(r(x, y)\) is the pixel’s region number. \(i_d(x, y)\) is 1 or 0: it is set to 1 if the pixel located at \((x, y)\) is at least \(d = 2\) pixels away from any region boundary, otherwise, it is set to 0.

Given these weights, Sundberg et al. [44] use a weighted least squares calculation to fit the optical flow (the \(u\) and \(v\) components) as functions of \(x\) and \(y\) pixel values shown in the two equations given below:

\[
\begin{align*}
    e^R_u &= \sum_{(x, y)} w^R_i(x, y)(A^R u x + B^R u y + C^R - u(x, y))^2 \\
    \end{align*}
\]

(4.7)

and

\[
\begin{align*}
    e^R_v &= \sum_{(x, y)} w^R_i(x, y)(A^R v x + B^R v y + C^R - v(x, y))^2.
    \end{align*}
\]

(4.8)

We want the least squares calculation of \(u\) and \(v\) such that \(e^R_u\) and \(e^R_v\) are minimized (optimally to 0). Of course, the \(u\) and \(v\) values are computed using Brox et al.’s global optical flow algorithm.

We give the details for the minimization of Equations (4.7) and (4.8). To minimize the
values of $e_u^R$ and $e_v^R$, we need to compute $A_u^R$, $B_u^R$ and $C_u^R$ and $A_v^R$, $B_v^R$ and $C_v^R$. In order to get the minimize these equations, we need to calculate their partial derivatives:

\[
\begin{align*}
\frac{\partial e_u^R}{\partial \lambda_u} &= 2 \sum_{(x,y)} w_i^R(x,y) x (A_u^R x + B_u^R y + C_u^R - u(x,y)) \\
\frac{\partial e_u^R}{\partial B_u} &= 2 \sum_{(x,y)} w_i^R(x,y) y (A_u^R x + B_u^R y + C_u^R - u(x,y)) \\
\frac{\partial e_u^R}{\partial C_u} &= 2 \sum_{(x,y)} w_i^R(x,y) (A_u^R x + B_u^R y + C_u^R - u(x,y)) 
\end{align*}
\]

and

\[
\begin{align*}
\frac{\partial e_v^R}{\partial \lambda_v} &= 2 \sum_{(x,y)} w_i^R(x,y) x (A_v^R x + B_v^R y + C_v^R - v(x,y)) \\
\frac{\partial e_v^R}{\partial B_v} &= 2 \sum_{(x,y)} w_i^R(x,y) y (A_v^R x + B_v^R y + C_v^R - v(x,y)) \\
\frac{\partial e_v^R}{\partial C_v} &= 2 \sum_{(x,y)} w_i^R(x,y) (A_v^R x + B_v^R y + C_v^R - v(x,y)) 
\end{align*}
\]

Setting $\frac{\partial e_u^R}{\partial \lambda_u} = 0$, $\frac{\partial e_u^R}{\partial B_u} = 0$ and $\frac{\partial e_u^R}{\partial C_u} = 0$, $\frac{\partial e_v^R}{\partial \lambda_v} = 0$, $\frac{\partial e_v^R}{\partial B_v} = 0$, $\frac{\partial e_v^R}{\partial C_v} = 0$ to minimum Equations (4.9) and (4.10). Then We distribute the terms and transform these equations into matrix form as:

\[
\begin{pmatrix}
\sum_{(x,y)} w_i^R(x,y) x^2 & \sum_{(x,y)} w_i^R(x,y) xy & \sum_{(x,y)} w_i^R(x,y) x \\
\sum_{(x,y)} w_i^R(x,y) xy & \sum_{(x,y)} w_i^R(x,y) y^2 & \sum_{(x,y)} w_i^R(x,y) y \\
\sum_{(x,y)} w_i^R(x,y) x & \sum_{(x,y)} w_i^R(x,y) y & \sum_{(x,y)} w_i^R(x,y)
\end{pmatrix}
\begin{pmatrix}
A_u^R \\
B_u^R \\
C_u^R
\end{pmatrix} =
\begin{pmatrix}
\sum_{(x,y)} w_i^R(x,y) x u(x,y) \\
\sum_{(x,y)} w_i^R(x,y) y u(x,y) \\
\sum_{(x,y)} w_i^R(x,y) u(x,y)
\end{pmatrix}
\]

and

\[
\begin{pmatrix}
\sum_{(x,y)} w_i^R(x,y) x^2 & \sum_{(x,y)} w_i^R(x,y) xy & \sum_{(x,y)} w_i^R(x,y) x \\
\sum_{(x,y)} w_i^R(x,y) xy & \sum_{(x,y)} w_i^R(x,y) y^2 & \sum_{(x,y)} w_i^R(x,y) y \\
\sum_{(x,y)} w_i^R(x,y) x & \sum_{(x,y)} w_i^R(x,y) y & \sum_{(x,y)} w_i^R(x,y)
\end{pmatrix}
\begin{pmatrix}
A_v^R \\
B_v^R \\
C_v^R
\end{pmatrix} =
\begin{pmatrix}
\sum_{(x,y)} w_i^R(x,y) x v(x,y) \\
\sum_{(x,y)} w_i^R(x,y) y v(x,y) \\
\sum_{(x,y)} w_i^R(x,y) v(x,y)
\end{pmatrix}
\]

We can write these two matrix equations as $L_u x_u = R_u$ and $L_v x_v = R_v$. We can solve these systems of equations using the left division operator in Matlab:

$x_u = L\_u \backslash R\_u$
to compute the parameters \( x_u \) and \( x_v \). Having \( A_u^R, B_u^R, C_u^R \) and \( A_v^R, B_v^R, C_v^R \), we can re-estimate the optical flow as:

\[
(u_i^R(x, y), v_i^R(x, y)) = (A_{ui}^R x + B_{ui}^R y + C_{ui}^R, A_{vi}^R x + B_{vi}^R y + C_{vi}^R)
\]

(4.13)

Note that \( (u_i^R(x, y), v_i^R(x, y)) \) are significantly more accurate than the Brox et al. flow it replaces.

The final step is to compare the optical flow for the 2 adjacent regions at an occlusion edge point itself, we get two flow vectors, \( w_i^+ = (u_i^+, v_i^+) \) and \( w_i^- = (u_i^-, v_i^-) \). Intuitively, occlusion boundary pixels should have obviously different values for the two different flows. Using the equation for a variable \( \delta \) defined in Sundberg et al’s paper [44] we have:

\[
\delta = |(u_i^+ - u_i^-, v_i^+ - v_i^-)|
\]

(4.14)

We can also take advantage of the \( gPb \) output by combining motion feature \( \delta \) and \( gPb \) using:

\[
f = \rho \ast \delta + (1 - \rho) \ast gPb
\]

(4.15)

with \( \rho = 0.7 \). This weight \( \rho \) value is determined using machine learning (using a Support Vector Machine, SVM) as noted by Sundberg et al. [44]. Figure 4.9 shows the rocking horse boundary images from UCM on \( gPb + mg \), \( \delta \) and and \( f \).

Figure 4.9: (a) is boundary image generated by thresholding the UCM output on \( gPb + mg \), (b) motion feature image using \( \delta \) Equation (4.14) and (c) \( f \) image using Equation (4.15).
The next step is boundary assignment, we want to mark which object the occlusion boundary belongs to. [This step is not needed for our optical flow calculation.] Sundberg et al. offers two equations to make this decision. These equations measure the distances between the motion of the edge and each of the two neighboring regions. Calling the two regions $+$ and $-$, we compute:

$$
\delta^+ = \frac{1}{n} \sum_i^n \sqrt{(u^+_i - u^0_i)^2 + (v^+_i - v^0_i)^2}
$$

(4.16)

$$
\delta^- = \frac{1}{n} \sum_i^n \sqrt{(u^-_i - u^0_i)^2 + (v^-_i - v^0_i)^2}
$$

(4.17)

where $n$ is the number of edge pixels on this edge fragment. Sundberg et al. say “it is natural to simply consider the optical flow on the edge to obtain the edge motion as $(u^0_i, v^0_i)$ at each edge pixel $i$”. Further, “since the variational optical flow is most accurate on edges” they use the Brox et al. optical flow as $(u^0_i, v^0_i)$. We assign the boundary pixel to the region with the smaller distance. We need to extract boundary fragments from our boundary detector output. We divide all boundaries into some independent continual boundaries. $n$ is the number of the edge pixels on every boundary fragment. The algorithm checks every pixel on the fragment, using these two equations to calculate two distances between the edge pixel and the two neighboring regions. By comparing these two distances values, we assign the pixel to the region with the nearer distance. We present boundary assignment results in Figure 4.10.

We present more results using this approach shown in Figures 4.11, 4.12, 4.13 and 4.14 for the tennis and bench sequences.

In order to compare the boundary detection results with and without the motion gradient, we also produce these outputs just using the $gPb$ detector, shown in Figures 4.15 and 4.16 for the rocking horse image, in Figures 4.17 and 4.18 for tennis image and in Figures 4.19 and 4.20 for the bench image.

For boundaries in Figures 4.16, 4.18, 4.18 and 4.20 we extract boundary fragments by looking at every unvisited boundary pixel in a 8-connected neighbourhood. We mark all pixels in the same fragment with the same index number. Otherwise, we will treat that pixel as start of another boundary fragment. We can see that these outputs that using $mg$ produced much
Figure 4.10: (a) is region map generated by our floodfill algorithm colored in Matlab jet(255) colormap using the occlusion boundary computed by $gPb+mg$ for the rocking horse. We color the boundaries as white while the different regions have different (non-white)colors. (b) This boundary detection result is generated by using Equations (4.16) and (4.17). We assign boundaries to the regions with the smaller distance and color it in that region’s color.

Figure 4.11: (a) is boundary image generated by computing the UCM output on $gPb+mg$, (b) the motion feature image using $\delta$ given in Equation (4.14) and (c) is the $f$ image using Equation 4.15.

better and cleaner boundaries.

4.6.2 Estimation of Optical Flow Using Occlusion Boundaries and Closed Regions

Our idea for re-estimation of optical flow is derived from the Sundberg et al. paper. In their algorithm, they utilize optical flow to do motion detection for occlusion boundary detection, as we described above. It is noteworthy that they define some equations to generate optical
Figure 4.12: (a) is region map generated by our floodfill algorithm colored in the Matlab jet(255) colormap using the boundary computed by $gPb+mg$. We color the boundaries white while coloring the regions non-white. (b) This result is generated using Equations (4.16) and (4.17). We assign boundaries to regions with the smaller distance and color it in that region’s color.

Figure 4.13: (a) is boundary image generated by computing the UCM output on $gPb+mg$, (b) the motion feature image using $\delta$ given in Equation (4.14) and (c) is the $f$ image using Equation (4.15).

flow, but they don’t show any optical flow in their paper (after all, the paper is about detecting occlusion boundaries and only used optical flow to do this). Our goal is to estimate optical flow in the closed regions bounded by occlusion edges. In this way, optical flow does not bleed across occlusion boundaries.

We start the process by computing the boundary from UCM using $gPb$ or $gPb+mg$ as inputs. Then we use our floodfill algorithm to get the region map. Figure 4.21 shows this calculation for the rocking horse image.
Figure 4.14: (a) is region map generated by our floodfill algorithm colored in the Matlab jet(255) colormap using the boundary computed by \( gPb+mg \). We color the boundaries white while coloring the regions non-white. (b) This result is generated using Equations (4.16) and (4.17). We assign boundaries to regions with the smaller distance and color it in that region’s color.

Figure 4.15: (a) is boundary image generated by computing the UCM output based on \( gPb \), (b) the motion feature image using \( \delta \) given by Equation (4.14) and (c) the \( f \) image using Equation (4.15).

### 4.6.3 1st Order Estimation

We can use an affine (1st order model) to estimate optical flow within a region. Affine transformations include contraction, expansion, dilation, reflection, rotation, shear, rotation, dilation and translation and their combinations. This model handles divergence/contraction, shear and deformation of optical flow. We use the following steps:

1. For each region, we use all pixels inside this region to calculate the parameters \( A_{\mu}^R, B_{\mu}^R \)
Figure 4.16: (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by gPb. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color.

Figure 4.17: (a) is boundary image generated by computing the UCM output based on gPb, (b) the motion feature image using δ given by Equation (4.14) and (c) the f image using Equation (4.15).

and $C^R_u$ and $A^R_v$, $B^R_v$ and $C^R_v$ via least squares We use Equations (4.20) and (4.21): these involve the multiplication of a $n \times 3$ matrix by a $3 \times 1$ matrix being equal to a $n \times 1$ matrix, which consists of $u$ and $v$ values produced by Brox et al.’s algorithm. Since:

$$A^R_u x + B^R_u y + C^R_u = u(x, y)$$

(4.18)

$$A^R_v x + B^R_v y + C^R_v = v(x, y)$$

(4.19)
Figure 4.18: (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by gPb. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color.

Figure 4.19: (a) is boundary image generated by computing the UCM output based on gPb, (b) the motion feature image using δ given by Equation (4.14) and (c) the f image using Equation (4.15).

we can write (in matrix form):

\[
\begin{pmatrix}
  x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 \\
  \vdots & \vdots & \vdots \\
  x_n & y_n & 1
\end{pmatrix}
\begin{pmatrix}
  A^R_u \\
  B^R_u \\
  C^R_u
\end{pmatrix} =
\begin{pmatrix}
  u(x,y) \\
  u(x,y) \\
  \vdots \\
  u(x,y)
\end{pmatrix}
\]

(4.20)
Figure 4.20: (a) is region map generated by our floodfill algorithm, colored by the Matlab jet(255) colormap using the boundary computed by gPb. We color the boundaries white while different regions are colored in different (non-white) colors. (b) is the result generated by using Equations (4.16) and (4.17), we assign boundaries into regions with the smaller distance and color it in that region’s color.

Figure 4.21: (a) is the computed UCM boundary while (b) is the region map generated by our floodfill algorithm, colored using the Matlab jet(255) colormap using that boundary. We color the boundaries white while different regions are colored with different (non-white) colors.

and

\[
\begin{bmatrix}
  x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 \\
  \vdots & \vdots & \vdots \\
  x_n & y_n & 1 \\
\end{bmatrix}
\begin{bmatrix}
  A_v^R \\
  B_v^R \\
  C_v^R \\
\end{bmatrix}
= 
\begin{bmatrix}
  v(x, y) \\
  v(x, y) \\
  \vdots \\
  v(x, y) \\
\end{bmatrix}.
\]  
(4.21)
2. We use these parameters to calculate new optical flow using:

\[ u'(x, y) = A_u^R x + B_u^R y + C_u^R \]  \hspace{1cm} (4.22)

and

\[ v'(x, y) = A_v^R x + B_v^R y + C_v^R. \]  \hspace{1cm} (4.23)

We demonstrate our optical flow re-estimation by showing the Brox et al. optical flow and our new optical flow result for the rocking horse image in Figure 4.22. The new optical flow was calculated using Equations (4.22) and (4.23). We also color code the optical flow vectors with the same colors as the corresponding regions. Moreover, in order to show our new optical flow clearly, we overlay the new optical flow on the region image to show optical flow in different regions. We also show the new optical flow on the occlusion boundaries. This all is shown in Figure 4.23, where the original image or the regions map is used as the background.

Figure 4.22: (a) is the optical flow estimation using Brox et al. while (b) is new estimated optical flow using our 1st order method. We show the optical flow in every region in different colors. We can see that the affine model result is a smooth optical flow for each region.
Figure 4.23: (a) is new estimated optical flow using our 1st order method overlaid on the original image, with white pixels indicating the detected boundaries. (b) is new optical flow at the boundaries colored with the regions’ colors they belong to with white pixels indicating the detected boundaries. (c) and (d) show the same flow fields as in (a) and (b) but they are overlaid on the colored regions map.

### 4.6.4 2nd Order Estimation

We also use a 2nd order model to estimate optical flow. This model handles planar surfaces (at any orientation), saddle point surfaces and spherical surfaces. For each region, we use the same setup to estimate the flow as for the 1st order method in the section above. The equations we solve are different but the steps are the same:

1. For each region, all pixels inside this region will be took into account. This time we use quadratic equations to estimate optical flow. We use 2nd order derivatives to calculate the
parameters $A_u^R$, $B_u^R$, $C_u^R$, $D_u^R$, $E_u^R$, and $F_u^R$, which are given in Equations (4.26) and (4.27a below. These involve a $n \times 6$ matrix multiplied by a $6 \times 1$ matrix being equal to a $n \times 1$ matrix, which consists of $u$ and $v$ produced by the Brox et al. algorithm.

$$A_u^R x + B_u^R y + C_u^R + D_u^R x^2 + E_u^R x y + F_u^R y^2 = u(x, y) \quad (4.24)$$

and

$$A_v^R x + B_v^R y + C_v^R + D_v^R x^2 + E_v^R x y + F_v^R y^2 = v(x, y) \quad (4.25)$$

We can write these equations in matrix form as:

$$\begin{pmatrix}
  x_1 & y_1 & 1 & x_1^2 & x_1 y_1 & y_1^2 \\
  x_2 & y_2 & 1 & x_2^2 & x_2 y_2 & y_2^2 \\
  ... & ... & ... & ... & ... & ... \\
  x_n & y_n & 1 & x_n^2 & x_n y_n & y_n^2
\end{pmatrix} \begin{pmatrix}
  A_u^R \\
  B_u^R \\
  C_u^R \\
  D_u^R \\
  E_u^R \\
  F_u^R
\end{pmatrix} = \begin{pmatrix}
  u(x, y) \\
  u(x, y) \\
  ... \\
  u(x, y)
\end{pmatrix} \quad (4.26)$$

and

$$\begin{pmatrix}
  x_1 & y_1 & 1 & x_1^2 & x_1 y_1 & y_1^2 \\
  x_2 & y_2 & 1 & x_2^2 & x_2 y_2 & y_2^2 \\
  ... & ... & ... & ... & ... & ... \\
  x_n & y_n & 1 & x_n^2 & x_n y_n & y_n^2
\end{pmatrix} \begin{pmatrix}
  A_v^R \\
  B_v^R \\
  C_v^R \\
  D_v^R \\
  E_v^R \\
  F_v^R
\end{pmatrix} = \begin{pmatrix}
  v(x, y) \\
  v(x, y) \\
  ... \\
  v(x, y)
\end{pmatrix}. \quad (4.27)$$

2. We use these parameters to calculate our new optical flow using:

$$u(x, y) = A_u^R x + B_u^R y + C_u^R + D_u^R x^2 + E_u^R x y + F_u^R y^2 \quad (4.28)$$
and

\[ v(x, y) = A^R_v x + B^R_v y + C^R_v + D^R_v x^2 + E^R_v xy + F^R_v y^2. \] (4.29)

We show the 2nd order optical flow for the rocking horse image. Figure 4.24 shows the new optical flow vectors both uncolored and colored (by region color). Figure 4.25 overlays new optical flow vectors on the original image and the regions map. The optical flow vectors are colored in their regions’ colors. The boundaries pixels are colored in white.

Figure 4.24: (a) and (b) are new estimated optical flow using our 1st and 2nd order method while every region colored by region color so as to observe the performance inside regions.
Figure 4.25: (a) is new estimated optical flow using our 2nd order method overlaid on the original image with white pixels indicating the detected boundaries. (b) is new optical flow at the boundaries boundaries, colored by the regions’ colors that they belong to, again with white pixels indicating the detected boundary pixels. (c) and (d) are (a) and (b) but with optical flow overlaid on the colored regions map.
Chapter 5

Experimental Results

We have tested our algorithms on various image sequences. Figures 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6 show the boundary, the region map, the 1\textsuperscript{st} and 2\textsuperscript{nd} order optical flows, the colored optical flow and the optical flow at the boundaries and regions overlaid on original images and regions map. These figures show this information for the tennis and bench images.

![Figure 5.1](image)

Figure 5.1: (a) is boundary image output by using the UCM algorithm for the tennis image and (b) is regions map generated by our floodfill algorithm.

As the reader can see, we have various methods to compute optical flow. We need a method to evaluate which algorithm is the best and how each algorithm performs. Even though we can qualitatively judge which optical flow result is better by simply looking at all results However,
Figure 5.2: (a) is colored Brox et al. optical flow for the tennis image. (b) and (c) is the 1st order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 1st order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white.
Figure 5.3: (a) is colored Brox et al. optical flow for the tennis image. (b) and (c) is the 2\textsuperscript{nd} order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 2\textsuperscript{nd} order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white.
we also need to perform a quantitative analysis. This is especially important when there are only minor differences in the compute optical flows.

5.1 Qualitative Analysis

Estimating optical flow using a variational approach is considered the best approach for computing optical flow, but the approach has some drawbacks. As we can clearly see in our flow results, optical flow is a mess at occlusion boundary pixels and some parts inside the closed occlusion regions near the boundaries.

We hypothesize that the use of the 1\textsuperscript{st} and 2\textsuperscript{nd} order would improve the accuracy of optical flow in occlusion regions. We believe that pixels in the same region should have common motion. We can clearly see that after this re-estimation of optical flow, performed independently in each occlusion region, our re-estimated optical flow looks much better and now varies smoothly everywhere, as optical flow at occlusion boundaries are not considered in the calculation. Hence, there is no bleeding of flow across occlusion boundaries as in, say, global Brox et al. optical flow. The optical flow images display optical flow vectors in every region.

Even though both the 1\textsuperscript{st} and 2\textsuperscript{nd} models return good estimation of region optical flow,
intuitively, we might think that 2\textsuperscript{nd} order optical flow should perform better than 1\textsuperscript{st} order optical flow. But we see few (if any) differences when comparing the two outputs visually; they look the same. So a quantitative metric is needed to show their difference. Also it is good to know whether and how much these two methods improve the optical flow performance over the original Brox et al. optical flow.

5.2 Quantitative Analysis

One idea for performing quantitative analysis when the correct optical flow is unknown is to construct first image from the second image using the computed optical flow. Then we can compute warping error as the root mean squared (RMS) difference between the original image and the warped image. This difference has error due to either the interpolation method used or to errors in the optical flow. This type of error was investigated in Lin and Barron [29]. Below, we give some MatLab code to do this warping. Given image arguments, \texttt{img1} and \texttt{img2}, and optical flow arguments, \texttt{u} and \texttt{v}, we compute the \texttt{warped\_image} (which should be \texttt{img1} is \texttt{u} and \texttt{v} are perfect) using bilinear interpolation. Below is MatLab that will do this.

```matlab
% return the warping error, the error image and the warped image
function [rms_warping_error,error_image,warped_image]=warp2D(img1,img2,u,v)
[height,width] = size(img2)
[X,Y]=meshgrid([1:width],[1:height]);
% x and y coordinates in the second image corresponding to X and Y in the first image
xnew=X+u;
ynew=Y+v;

% take care of image boundaries
xnew(xnew>width)=width;
xnew(xnew<1)=1;
ynew(ynew>height)=height;
ynew(ynew<1)=1;

warped_image=interp2(X,Y,img2,xnew,ynew,'bilinear');
error_image=abs(img1-warped_image);
% warping error is the root mean squared of the difference
```

% return the warping error, the error image and the warped image
function [rms_warping_error,error_image,warped_image]=warp2D(img1,img2,u,v)
[height,width] = size(img2)
[X,Y]=meshgrid([1:width],[1:height]);
% x and y coordinates in the second image corresponding to X and Y in the first image
xnew=X+u;
ynew=Y+v;

% take care of image boundaries
xnew(xnew>width)=width;
xnew(xnew<1)=1;
ynew(ynew>height)=height;
ynew(ynew<1)=1;

warped_image=interp2(X,Y,img2,xnew,ynew,'bilinear');
error_image=abs(img1-warped_image);
% warping error is the root mean squared of the difference
### Chapter 5. Experimental Results

<table>
<thead>
<tr>
<th>Image</th>
<th>Brox et al.</th>
<th>1\textsuperscript{st} Order</th>
<th>2\textsuperscript{nd} Order</th>
<th>Zero flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rocking Horse 10\textsuperscript{th} image</td>
<td>0.0807</td>
<td>0.0439</td>
<td>0.0444</td>
<td>0.0818</td>
</tr>
<tr>
<td>Tennis 744\textsuperscript{th} image</td>
<td>0.1015</td>
<td>0.0693</td>
<td>0.0698</td>
<td>0.0762</td>
</tr>
<tr>
<td>Bench 10\textsuperscript{th} image</td>
<td>0.1036</td>
<td>0.0305</td>
<td>0.0327</td>
<td>0.1036</td>
</tr>
</tbody>
</table>

Table 5.1: We perform warping error calculation for: the 10\textsuperscript{th} image of the rocking horse sequence and bench sequences and the 744\textsuperscript{th} image of the tennis sequence for the Brox et al., 1\textsuperscript{st} order and 2\textsuperscript{nd} order flow fields. Zero flow means using vector (0,0) to compute warping errors as upper bound. Figure 5.7 shows the warping error images.

\[
\text{rms_warping_error} = \text{norm(error_image(:)/size(error_image,1)*size(error_image,2))};
\]

We have computed warping error for our optical flow algorithms’ output to show their performance: we report results for Brox et al. flow as well as our 1\textsuperscript{st} and 2\textsuperscript{nd} order models in Table 5.1. Also we show the reconstructed images in Figure 5.8 for the rocking horse, tennis and bench images for Brox et al. flow and for our 1\textsuperscript{st} and 2\textsuperscript{nd} order models. We also use zero flow (all pixels have velocities (0,0)) to demonstrate the maximum warping error for 2 images. Using zero flow, we warp the 2\textsuperscript{nd} image back into the 1\textsuperscript{st} image (of course the 2\textsuperscript{nd} images stays unchanged) and then we compute the warping error as just the difference between the 2 images. We hypothesize this to be the worst case. From Table 5.1 we see that both our methods perform better than the original Brox et al. optical flow but that 2\textsuperscript{nd} order method isn’t better than our 1\textsuperscript{st} order method. From Figure 5.8 we see that the Brox et al. reconstructed images have a little error along the images’ discontinuities (for example, along the bench’s occlusion boundary). We also note that qualitatively, the 1\textsuperscript{st} order and 2\textsuperscript{nd} order reconstructed images look identical. We perform Structural Similarity Index (SSIM) calculation on reconstructed images to measure image quality shown in Table 5.2, we calculate this using Matlab function.

Our main interest is in the optical flow at occlusion boundaries. So we choose a region of interest (delimited in red) so as to compare their warping errors. Numeric values are given Table 5.3 while the images are shown in Figure 5.9. For the bench image, we see a significant improvement between the Brox et al. optical flow and our re-estimated optical flow. The rocking horse and tennis show only slight improvements for our re-estimated optical flow over
**Table 5.2:** We perform Structural Similarity Index (SSIM) calculation for: the 10th image of the rocking horse sequence and bench sequences and the 744th image of the tennis sequence for the Brox et al., 1st order and 2nd order flow fields.

<table>
<thead>
<tr>
<th>Image</th>
<th>Brox et al.</th>
<th>1st Order</th>
<th>2nd Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rocking Horse 10th image</td>
<td>0.6684</td>
<td>0.8859</td>
<td>0.8856</td>
</tr>
<tr>
<td>Tennis 744th image</td>
<td>0.5860</td>
<td>0.8521</td>
<td>0.8508</td>
</tr>
<tr>
<td>Bench 10th image</td>
<td>0.5243</td>
<td>0.9424</td>
<td>0.9437</td>
</tr>
</tbody>
</table>

**Table 5.3:** We perform warping error calculation for 3 red rectangles in: the 10th image of the rocking horse sequence and bench sequences and the 744th image of the tennis sequence for the Brox et al., 1st order and 2nd order flow fields. Zero flow means we use vector (0,0) to compute warping errors as upper bound.

<table>
<thead>
<tr>
<th>Image</th>
<th>Brox et al</th>
<th>1st Order</th>
<th>2nd Order</th>
<th>Zero flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse 10th image</td>
<td>0.0807</td>
<td>0.0434</td>
<td>0.0432</td>
<td>0.0896</td>
</tr>
<tr>
<td>Tennis 744th image</td>
<td>0.1395</td>
<td>0.1223</td>
<td>0.1225</td>
<td>0.1298</td>
</tr>
<tr>
<td>Bench 10th image</td>
<td>0.0901</td>
<td>0.0254</td>
<td>0.0251</td>
<td>0.0399</td>
</tr>
</tbody>
</table>

For the tennis sequence, it is difficult to tell which of the 1st and 2nd order flows if the best from a visual inspection of a movie of the image motions. Things appear to be moving left and down. There are shadows from the moving ball, the tennis player and the fence. Our warping error metric on the red boxed area as shown in Figure 5.10 seems to indicate the 2nd order flow is a little better than the 1st order flow. Note that the ground is a reasonably uniform area with little texture, other than the player’s feet (so most flow fields might return good warping error measures). These flows for this sequence show a weakness with the use of the warping error metric (we did look at SSIM error as a possible alternative but we could draw no definite conclusions).

We also evaluate individual regions’ warping errors to evaluate their performance in regions. Our 1st and 2nd order optical flow methods are the same and they both exhibit some improvement based on Brox et al.’s optical flow method. As there are too many regions in the tennis image, it is a problem to show each region’s warping errors in an organized way. Listing the warping error by region number in a table may not be the best way to show this information.
tion. But it is still a good way to show regions’ warping errors when there are not too many regions in the image. Tables 5.4 and 5.5 and Figures 5.11 and 5.12 show the warping error for each region in the rocking horse and bench images. We can see that individual region optical flows are about 20% to 50% better for the 1st and 2nd optical flow methods over the Brox et al. optical flow method. Table 5.6 show the warping error for all regions in tennis image. Our methods are better than Brox et al.’s method in all regions. Pixels percentage number indicates how many pixels in each region, we show them as percentage in the overall image.

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.’s</th>
<th>1st Order</th>
<th>2nd Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.1003</td>
<td>0.0454</td>
<td>0.0468</td>
<td>10.47</td>
</tr>
<tr>
<td>3</td>
<td>0.1227</td>
<td>0.0459</td>
<td>0.0447</td>
<td>0.30</td>
</tr>
<tr>
<td>4</td>
<td>0.1483</td>
<td>0.0321</td>
<td>0.0497</td>
<td>0.37</td>
</tr>
<tr>
<td>5</td>
<td>0.0788</td>
<td>0.0251</td>
<td>0.0243</td>
<td>1.89</td>
</tr>
<tr>
<td>6</td>
<td>0.1763</td>
<td>0.0500</td>
<td>0.0370</td>
<td>0.30</td>
</tr>
<tr>
<td>7</td>
<td>0.1477</td>
<td>0.0679</td>
<td>0.0454</td>
<td>0.07</td>
</tr>
<tr>
<td>8</td>
<td>0.0699</td>
<td>0.0237</td>
<td>0.0218</td>
<td>2.26</td>
</tr>
<tr>
<td>9</td>
<td>0.0577</td>
<td>0.0421</td>
<td>0.0420</td>
<td>39.03</td>
</tr>
<tr>
<td>10</td>
<td>0.0232</td>
<td>0.0101</td>
<td>0.0098</td>
<td>0.71</td>
</tr>
<tr>
<td>11</td>
<td>0.0870</td>
<td>0.0388</td>
<td>0.0459</td>
<td>0.40</td>
</tr>
<tr>
<td>12</td>
<td>0.0752</td>
<td>0.0186</td>
<td>0.0192</td>
<td>0.50</td>
</tr>
<tr>
<td>13</td>
<td>0.1463</td>
<td>0.0340</td>
<td>0.0442</td>
<td>1.55</td>
</tr>
<tr>
<td>14</td>
<td>0.3105</td>
<td>0.1235</td>
<td>0.1248</td>
<td>0.74</td>
</tr>
<tr>
<td>15</td>
<td>0.0611</td>
<td>0.0189</td>
<td>0.0171</td>
<td>1.31</td>
</tr>
<tr>
<td>16</td>
<td>0.2424</td>
<td>0.1278</td>
<td>0.1656</td>
<td>0.16</td>
</tr>
<tr>
<td>17</td>
<td>0.0719</td>
<td>0.0285</td>
<td>0.0299</td>
<td>10.68</td>
</tr>
<tr>
<td>18</td>
<td>0.0599</td>
<td>0.0232</td>
<td>0.0214</td>
<td>0.60</td>
</tr>
<tr>
<td>19</td>
<td>0.1868</td>
<td>0.1437</td>
<td>0.1510</td>
<td>0.17</td>
</tr>
<tr>
<td>20</td>
<td>0.0478</td>
<td>0.0186</td>
<td>0.0197</td>
<td>12.35</td>
</tr>
<tr>
<td>21</td>
<td>0.1662</td>
<td>0.0819</td>
<td>0.0669</td>
<td>0.45</td>
</tr>
<tr>
<td>22</td>
<td>0.0819</td>
<td>0.0287</td>
<td>0.0290</td>
<td>13.95</td>
</tr>
<tr>
<td>23</td>
<td>0.0865</td>
<td>0.0341</td>
<td>0.0535</td>
<td>0.031</td>
</tr>
</tbody>
</table>

Table 5.4: Region warping errors calculated by using Brox et al., 1st order and 2nd order methods for the 10th rocking horse image.

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.’s</th>
<th>1st Order</th>
<th>2nd Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0892</td>
<td>0.0320</td>
<td>0.0319</td>
<td>1.60</td>
</tr>
<tr>
<td>3</td>
<td>0.1629</td>
<td>0.0713</td>
<td>0.0728</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>4</td>
<td>0.1146</td>
<td>0.0973</td>
<td>0.0979</td>
<td>5.64</td>
</tr>
<tr>
<td>5</td>
<td>0.0499</td>
<td>0.0166</td>
<td>0.0171</td>
<td>5.60</td>
</tr>
<tr>
<td>6</td>
<td>0.1831</td>
<td>0.0774</td>
<td>0.0857</td>
<td>0.07</td>
</tr>
<tr>
<td>7</td>
<td>0.0833</td>
<td>0.0305</td>
<td>0.0292</td>
<td>0.85</td>
</tr>
<tr>
<td>8</td>
<td>0.1519</td>
<td>0.0579</td>
<td>0.0818</td>
<td>0.24</td>
</tr>
<tr>
<td>9</td>
<td>0.0645</td>
<td>0.0260</td>
<td>0.0271</td>
<td>0.68</td>
</tr>
<tr>
<td>10</td>
<td>0.0821</td>
<td>0.0278</td>
<td>0.0282</td>
<td>1.54</td>
</tr>
<tr>
<td>11</td>
<td>0.0638</td>
<td>0.0243</td>
<td>0.0244</td>
<td>0.83</td>
</tr>
<tr>
<td>12</td>
<td>0.1660</td>
<td>0.0574</td>
<td>0.0907</td>
<td>0.07</td>
</tr>
<tr>
<td>13</td>
<td>0.2301</td>
<td>0.0930</td>
<td>0.0943</td>
<td>0.15</td>
</tr>
<tr>
<td>14</td>
<td>0.1713</td>
<td>0.0819</td>
<td>0.0746</td>
<td>0.26</td>
</tr>
<tr>
<td>15</td>
<td>0.0925</td>
<td>0.0299</td>
<td>0.0297</td>
<td>0.68</td>
</tr>
<tr>
<td>16</td>
<td>0.0630</td>
<td>0.0249</td>
<td>0.0250</td>
<td>2.86</td>
</tr>
<tr>
<td>17</td>
<td>0.1260</td>
<td>0.0418</td>
<td>0.0459</td>
<td>0.08</td>
</tr>
<tr>
<td>18</td>
<td>0.1256</td>
<td>0.0417</td>
<td>0.0433</td>
<td>0.50</td>
</tr>
<tr>
<td>19</td>
<td>0.1683</td>
<td>0.0935</td>
<td>0.0761</td>
<td>0.06</td>
</tr>
<tr>
<td>20</td>
<td>0.0948</td>
<td>0.0342</td>
<td>0.0381</td>
<td>0.48</td>
</tr>
<tr>
<td>21</td>
<td>0.0702</td>
<td>0.0587</td>
<td>0.0556</td>
<td>0.02</td>
</tr>
<tr>
<td>22</td>
<td>0.1057</td>
<td>0.0347</td>
<td>0.0362</td>
<td>0.42</td>
</tr>
<tr>
<td>23</td>
<td>0.1430</td>
<td>0.0828</td>
<td>0.0735</td>
<td>0.23</td>
</tr>
<tr>
<td>24</td>
<td>0.1502</td>
<td>0.0436</td>
<td>0.0527</td>
<td>0.06</td>
</tr>
<tr>
<td>25</td>
<td>0.1318</td>
<td>0.0613</td>
<td>0.0584</td>
<td>0.03</td>
</tr>
<tr>
<td>26</td>
<td>0.2061</td>
<td>0.0996</td>
<td>0.1254</td>
<td>0.02</td>
</tr>
<tr>
<td>27</td>
<td>0.1261</td>
<td>0.0303</td>
<td>0.0426</td>
<td>0.54</td>
</tr>
<tr>
<td>28</td>
<td>0.0796</td>
<td>0.0291</td>
<td>0.0304</td>
<td>0.19</td>
</tr>
<tr>
<td>29</td>
<td>0.1712</td>
<td>0.0770</td>
<td>0.0864</td>
<td>0.05</td>
</tr>
<tr>
<td>30</td>
<td>0.1335</td>
<td>0.0579</td>
<td>0.0545</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>31</td>
<td>0.1084</td>
<td>0.0362</td>
<td>0.0368</td>
<td>0.48</td>
</tr>
<tr>
<td>32</td>
<td>0.2408</td>
<td>0.2160</td>
<td>0.2179</td>
<td>0.06</td>
</tr>
<tr>
<td>33</td>
<td>0.0802</td>
<td>0.0310</td>
<td>0.0313</td>
<td>0.75</td>
</tr>
<tr>
<td>34</td>
<td>0.1457</td>
<td>0.0427</td>
<td>0.0492</td>
<td>0.21</td>
</tr>
<tr>
<td>35</td>
<td>0.1304</td>
<td>0.0479</td>
<td>0.0540</td>
<td>0.06</td>
</tr>
<tr>
<td>36</td>
<td>0.1602</td>
<td>0.1116</td>
<td>0.1279</td>
<td>0.03</td>
</tr>
<tr>
<td>37</td>
<td>0.1266</td>
<td>0.0349</td>
<td>0.0389</td>
<td>0.69</td>
</tr>
<tr>
<td>38</td>
<td>0.0824</td>
<td>0.0199</td>
<td>0.0263</td>
<td>0.07</td>
</tr>
<tr>
<td>39</td>
<td>0.0810</td>
<td>0.0604</td>
<td>0.0618</td>
<td>0.30</td>
</tr>
<tr>
<td>40</td>
<td>0.2508</td>
<td>0.2386</td>
<td>0.2422</td>
<td>0.31</td>
</tr>
<tr>
<td>41</td>
<td>0.1017</td>
<td>0.0317</td>
<td>0.0354</td>
<td>0.19</td>
</tr>
<tr>
<td>42</td>
<td>0.1535</td>
<td>0.0403</td>
<td>0.0389</td>
<td>0.07</td>
</tr>
<tr>
<td>43</td>
<td>0.1425</td>
<td>0.0865</td>
<td>0.0847</td>
<td>0.06</td>
</tr>
<tr>
<td>44</td>
<td>0.1517</td>
<td>0.0503</td>
<td>0.0557</td>
<td>0.17</td>
</tr>
<tr>
<td>45</td>
<td>0.1813</td>
<td>0.1136</td>
<td>0.1096</td>
<td>0.07</td>
</tr>
<tr>
<td>46</td>
<td>0.2871</td>
<td>0.2170</td>
<td>0.2447</td>
<td>0.02</td>
</tr>
<tr>
<td>47</td>
<td>0.1415</td>
<td>0.0657</td>
<td>0.0711</td>
<td>0.08</td>
</tr>
<tr>
<td>48</td>
<td>0.1274</td>
<td>0.0315</td>
<td>0.0407</td>
<td>0.60</td>
</tr>
<tr>
<td>49</td>
<td>0.0767</td>
<td>0.0779</td>
<td>0.0779</td>
<td>0.01</td>
</tr>
<tr>
<td>50</td>
<td>0.1378</td>
<td>0.0386</td>
<td>0.0384</td>
<td>0.19</td>
</tr>
<tr>
<td>51</td>
<td>0.1560</td>
<td>0.0762</td>
<td>0.0995</td>
<td>0.05</td>
</tr>
<tr>
<td>52</td>
<td>0.1005</td>
<td>0.0695</td>
<td>0.0680</td>
<td>0.28</td>
</tr>
<tr>
<td>53</td>
<td>0.0386</td>
<td>0.0170</td>
<td>0.0201</td>
<td>0.01</td>
</tr>
<tr>
<td>54</td>
<td>0.1522</td>
<td>0.0408</td>
<td>0.0427</td>
<td>0.67</td>
</tr>
<tr>
<td>55</td>
<td>0.1294</td>
<td>0.0345</td>
<td>0.0396</td>
<td>0.22</td>
</tr>
<tr>
<td>56</td>
<td>0.1578</td>
<td>0.1443</td>
<td>0.1582</td>
<td>0.15</td>
</tr>
<tr>
<td>57</td>
<td>0.0880</td>
<td>0.0295</td>
<td>0.0321</td>
<td>2.59</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>58</td>
<td>0.1399</td>
<td>0.0477</td>
<td>0.0479</td>
<td>0.07</td>
</tr>
<tr>
<td>59</td>
<td>0.0838</td>
<td>0.0303</td>
<td>0.0336</td>
<td>0.24</td>
</tr>
<tr>
<td>60</td>
<td>0.1146</td>
<td>0.0305</td>
<td>0.0311</td>
<td>0.70</td>
</tr>
<tr>
<td>61</td>
<td>0.0848</td>
<td>0.0278</td>
<td>0.0280</td>
<td>2.41</td>
</tr>
<tr>
<td>62</td>
<td>0.1342</td>
<td>0.0405</td>
<td>0.0447</td>
<td>0.18</td>
</tr>
<tr>
<td>63</td>
<td>0.0874</td>
<td>0.0312</td>
<td>0.0333</td>
<td>0.13</td>
</tr>
<tr>
<td>64</td>
<td>0.2328</td>
<td>0.2292</td>
<td>0.2281</td>
<td>0.32</td>
</tr>
<tr>
<td>65</td>
<td>0.0726</td>
<td>0.0211</td>
<td>0.0263</td>
<td>0.36</td>
</tr>
<tr>
<td>66</td>
<td>0.1082</td>
<td>0.0511</td>
<td>0.0519</td>
<td>0.19</td>
</tr>
<tr>
<td>67</td>
<td>0.1034</td>
<td>0.0396</td>
<td>0.0474</td>
<td>0.47</td>
</tr>
<tr>
<td>68</td>
<td>0.1005</td>
<td>0.0559</td>
<td>0.0578</td>
<td>0.77</td>
</tr>
<tr>
<td>69</td>
<td>0.0772</td>
<td>0.0356</td>
<td>0.0448</td>
<td>0.02</td>
</tr>
<tr>
<td>70</td>
<td>0.1770</td>
<td>0.0574</td>
<td>0.0736</td>
<td>0.05</td>
</tr>
<tr>
<td>71</td>
<td>0.2290</td>
<td>0.2206</td>
<td>0.2191</td>
<td>0.27</td>
</tr>
<tr>
<td>72</td>
<td>0.1443</td>
<td>0.0358</td>
<td>0.0364</td>
<td>0.42</td>
</tr>
<tr>
<td>73</td>
<td>0.1001</td>
<td>0.0854</td>
<td>0.0814</td>
<td>0.07</td>
</tr>
<tr>
<td>74</td>
<td>0.0678</td>
<td>0.0215</td>
<td>0.0210</td>
<td>0.74</td>
</tr>
<tr>
<td>75</td>
<td>0.0998</td>
<td>0.0787</td>
<td>0.0793</td>
<td>0.03</td>
</tr>
<tr>
<td>76</td>
<td>0.1424</td>
<td>0.0893</td>
<td>0.0980</td>
<td>0.06</td>
</tr>
<tr>
<td>77</td>
<td>0.1007</td>
<td>0.0286</td>
<td>0.0364</td>
<td>0.51</td>
</tr>
<tr>
<td>78</td>
<td>0.0735</td>
<td>0.0636</td>
<td>0.0706</td>
<td>0.02</td>
</tr>
<tr>
<td>79</td>
<td>0.0703</td>
<td>0.0301</td>
<td>0.0340</td>
<td>0.20</td>
</tr>
<tr>
<td>80</td>
<td>0.1159</td>
<td>0.0291</td>
<td>0.0348</td>
<td>0.89</td>
</tr>
<tr>
<td>81</td>
<td>0.0944</td>
<td>0.0305</td>
<td>0.0307</td>
<td>2.38</td>
</tr>
<tr>
<td>82</td>
<td>0.3030</td>
<td>0.3029</td>
<td>0.2962</td>
<td>0.22</td>
</tr>
<tr>
<td>83</td>
<td>0.1035</td>
<td>0.0240</td>
<td>0.0254</td>
<td>0.14</td>
</tr>
<tr>
<td>84</td>
<td>0.2222</td>
<td>0.2104</td>
<td>0.2117</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>85</td>
<td>0.1165</td>
<td>0.0680</td>
<td>0.0646</td>
<td>0.06</td>
</tr>
<tr>
<td>86</td>
<td>0.0919</td>
<td>0.0534</td>
<td>0.0536</td>
<td>5.31</td>
</tr>
<tr>
<td>87</td>
<td>0.1612</td>
<td>0.0394</td>
<td>0.0717</td>
<td>0.17</td>
</tr>
<tr>
<td>88</td>
<td>0.1333</td>
<td>0.0859</td>
<td>0.0883</td>
<td>0.09</td>
</tr>
<tr>
<td>89</td>
<td>0.1212</td>
<td>0.1090</td>
<td>0.1078</td>
<td>0.70</td>
</tr>
<tr>
<td>90</td>
<td>0.3039</td>
<td>0.3055</td>
<td>0.3055</td>
<td>0.45</td>
</tr>
<tr>
<td>91</td>
<td>0.0768</td>
<td>0.0302</td>
<td>0.0305</td>
<td>7.45</td>
</tr>
<tr>
<td>92</td>
<td>0.1163</td>
<td>0.0385</td>
<td>0.0500</td>
<td>0.07</td>
</tr>
<tr>
<td>93</td>
<td>0.1030</td>
<td>0.1006</td>
<td>0.0936</td>
<td>0.12</td>
</tr>
<tr>
<td>94</td>
<td>0.1711</td>
<td>0.1450</td>
<td>0.1569</td>
<td>0.01</td>
</tr>
<tr>
<td>95</td>
<td>0.2784</td>
<td>0.2675</td>
<td>0.2689</td>
<td>0.30</td>
</tr>
<tr>
<td>96</td>
<td>0.2035</td>
<td>0.2016</td>
<td>0.1905</td>
<td>0.05</td>
</tr>
<tr>
<td>97</td>
<td>0.0829</td>
<td>0.0281</td>
<td>0.0294</td>
<td>0.30</td>
</tr>
<tr>
<td>98</td>
<td>0.1587</td>
<td>0.1548</td>
<td>0.1513</td>
<td>0.35</td>
</tr>
<tr>
<td>99</td>
<td>0.3243</td>
<td>0.3667</td>
<td>0.3609</td>
<td>0.02</td>
</tr>
<tr>
<td>100</td>
<td>0.1499</td>
<td>0.1109</td>
<td>0.1236</td>
<td>0.57</td>
</tr>
<tr>
<td>101</td>
<td>0.2315</td>
<td>0.0768</td>
<td>0.1138</td>
<td>0.02</td>
</tr>
<tr>
<td>102</td>
<td>0.0635</td>
<td>0.0480</td>
<td>0.0463</td>
<td>4.59</td>
</tr>
<tr>
<td>103</td>
<td>0.1181</td>
<td>0.1053</td>
<td>0.0550</td>
<td>0.01</td>
</tr>
<tr>
<td>104</td>
<td>0.1617</td>
<td>0.1529</td>
<td>0.1542</td>
<td>0.64</td>
</tr>
<tr>
<td>105</td>
<td>0.0696</td>
<td>0.0694</td>
<td>0.0653</td>
<td>0.65</td>
</tr>
<tr>
<td>106</td>
<td>0.3931</td>
<td>0.3882</td>
<td>0.3959</td>
<td>0.04</td>
</tr>
<tr>
<td>107</td>
<td>0.3946</td>
<td>0.4196</td>
<td>0.4196</td>
<td>0.002</td>
</tr>
<tr>
<td>108</td>
<td>0.0455</td>
<td>0.0130</td>
<td>0.0131</td>
<td>2.59</td>
</tr>
<tr>
<td>109</td>
<td>0.0568</td>
<td>0.0159</td>
<td>0.0156</td>
<td>0.15</td>
</tr>
<tr>
<td>110</td>
<td>0.0649</td>
<td>0.0196</td>
<td>0.0182</td>
<td>0.13</td>
</tr>
<tr>
<td>111</td>
<td>0.0313</td>
<td>0.0097</td>
<td>0.0100</td>
<td>4.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>112</td>
<td>0.1627</td>
<td>0.0505</td>
<td>0.0883</td>
<td>0.05</td>
</tr>
<tr>
<td>113</td>
<td>0.0922</td>
<td>0.0286</td>
<td>0.0328</td>
<td>0.36</td>
</tr>
<tr>
<td>114</td>
<td>0.0719</td>
<td>0.0233</td>
<td>0.0224</td>
<td>1.82</td>
</tr>
<tr>
<td>115</td>
<td>0.1535</td>
<td>0.0336</td>
<td>0.0577</td>
<td>0.45</td>
</tr>
<tr>
<td>116</td>
<td>0.0678</td>
<td>0.0305</td>
<td>0.0261</td>
<td>0.44</td>
</tr>
<tr>
<td>117</td>
<td>0.0665</td>
<td>0.0421</td>
<td>0.0409</td>
<td>1.47</td>
</tr>
<tr>
<td>118</td>
<td>0.0332</td>
<td>0.0246</td>
<td>0.0239</td>
<td>9.96</td>
</tr>
<tr>
<td>119</td>
<td>0.0466</td>
<td>0.0366</td>
<td>0.0354</td>
<td>4.54</td>
</tr>
<tr>
<td>120</td>
<td>0.1549</td>
<td>0.1441</td>
<td>0.1663</td>
<td>0.01</td>
</tr>
<tr>
<td>121</td>
<td>0.3160</td>
<td>0.3347</td>
<td>0.3300</td>
<td>0.22</td>
</tr>
<tr>
<td>122</td>
<td>0.1453</td>
<td>0.0591</td>
<td>0.0603</td>
<td>0.16</td>
</tr>
<tr>
<td>123</td>
<td>0.1543</td>
<td>0.0677</td>
<td>0.0844</td>
<td>0.01</td>
</tr>
<tr>
<td>124</td>
<td>0.3051</td>
<td>0.2911</td>
<td>0.3044</td>
<td>0.02</td>
</tr>
<tr>
<td>125</td>
<td>0.0734</td>
<td>0.0698</td>
<td>0.0697</td>
<td>1.19</td>
</tr>
</tbody>
</table>

Table 5.6: Region warping errors calculated by using Brox et al., 1st order and 2nd order methods for the 744th tennis image.
### Chapter 5. Experimental Results

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.’s</th>
<th>1(^{st}) Order</th>
<th>2(^{nd}) Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.1181</td>
<td>0.0384</td>
<td>0.0380</td>
<td>59.88</td>
</tr>
<tr>
<td>3</td>
<td>0.0601</td>
<td>0.0156</td>
<td>0.0152</td>
<td>24.89</td>
</tr>
<tr>
<td>4</td>
<td>0.0828</td>
<td>0.0235</td>
<td>0.0236</td>
<td>11.32</td>
</tr>
<tr>
<td>5</td>
<td>0.1246</td>
<td>0.0373</td>
<td>0.0378</td>
<td>2.27</td>
</tr>
</tbody>
</table>

Table 5.5: Region warping errors calculated by using Brox et al., 1\(^{st}\) order and 2\(^{nd}\) order methods for the 10\(^{th}\) bench image.
Figure 5.5: (a) is colored Brox et al. optical flow for the bench image. (b) and (c) is the 1st order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 1st order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white.
Figure 5.6: (a) is colored Brox et al. optical flow for the bench image. (b) and (c) is the 2nd order estimated optical flow overlaid on the original image and on the regions image. (d) and (e) are 2nd order estimated optical flow at the boundaries displayed on the original image and regions map. The boundaries are colored white.
Figure 5.7: The left column images are warping error images of Brox et al.’s method, The middle column images are warping error images of 1\textsuperscript{st} order method, The middle column images are warping error images of 2\textsuperscript{nd} order method, They are computed on rocking horse, tennis and bench sequences.
Figure 5.8: The left column reconstructed images are the rocking horse, (a) is the original image, (d) is the Brox et al. reconstructed image, (g) is the 1st order reconstructed image and (j) is the 2nd order reconstructed image. (b), (e), (h) and (k) are the respective images for the tennis image while (c), (f), (i) and (l) are the respective images for the bench image.
Figure 5.9: (a), (d) and (g) show the rocking horse images in rectangle [80 80 480 460], (b), (e) and (h) show the tennis image in rectangle [70 180 360 360] and (c), (f) and (i) show the bench images in rectangle [1 100 190 320]. The top images are for Brox et al., the middle images are for the 1st order method and the bottom images are the 2nd order model. The rectangles in red indicate the region of interest. Each image shows the RMS warping error for that image.
Figure 5.10: We see some difference in our 1st and 2nd results of tennis image on the ground pats. Here we also show their difference in rectangle [250 1 360 360].
Figure 5.11: From top to bottom are the reconstructed images using Brox et al.’s method and our 1st and 2nd order method for each region’s warping errors displayed for the rocking horse image.
Figure 5.12: From top to bottom are the reconstructed images using Brox et al.’s method and our 1st and 2nd order method for each region’s warping errors displayed for the bench image.
Chapter 6

Conclusions and Future Work

6.1 Conclusion

We implemented Sundberg et al.’s occlusion boundary detection algorithm. Our implementation of the motion gradient ($mg$) proposed by Martin et al. may have a minor error but produces good results. We have shown qualitatively and quantitatively that simple $1^{st}$ and $2^{nd}$ order models imposed on the Brox et al. optical flow produce much better optical flow than the original Brox et al. optical flow. Re-estimation of the optical flow within a closed occlusion region is a good idea, provided the occlusion boundaries are accurate (as they appear to be). We do note the protrusion on one of the bench images.

6.2 Future Work

Developing addition optical flow methods for the occlusion regions is one possible avenue of future work. Also we can try to estimate optical flow based on sequences, by adding a temporal consistency constraint to the optical flow along the sequence, so flow varies smoothly over time.
Appendix A

A.1 Experimental results

Figure A.1 shows the UCM results, $gPb$, UCM using $gPb$ and UCM with $gPb$ with thresholding of 0.8, 0.8 and 0.4 respectively for the hand image sequence, the Hamburg taxi sequence and the trees sequence. Figure A.2 shows the segmentation results for the $gPb + mg$ algorithm, the UCM results using $gPb + mg$ and the UCM results using $gPb + gm$ with thresholding of 0.9, 0.9 and 0.6 for these 3 images. Figure A.3 shows Sundberg et al.’s results for 6th image of the hand sequence, Figure A.4 shows Sundberg et al.’s results for 10th image of the Hamburg taxi sequence while Figure A.5 shows Sundberg et al.’s results for 10th image of the the trees sequence. These 3 figures including $\delta, f$ and distance assignment to regions images.
Figure A.1: These are results output by the UCM algorithm. The top images are original images. The upper middle images are the output of the \textit{gPb} algorithm for $6^{th}$ image of hand image sequence, the $10^{th}$ image of the Hamburg taxi image sequence and $10^{th}$ image of the trees image sequence. The lower middle images are the UCM results using \textit{gPb}. The bottom images are the segmentation results for the UCM algorithm using \textit{gPb} with thresholding at 0.8, 0.8 and 0.4 for the 3 image sequences.
Figure A.2: Top images are the output for the $gPb + mg$ algorithm of 6th image of the hand images sequence, the 10th image of Hamburg taxi image sequence and of 10th image of the trees image sequence. The middle images are UCM results using $gPb + mg$ for the 3 images. The bottom image are the segmentation results of the UCM algorithm using $gPb + mg$ with thresholding at 0.9, 0.9 and 0.6 for the 3 images.
Figure A.3: Sundberg et al.’s results for 6th image the hand sequence, including $\delta$, $f$, region map and distance assignment to regions.
Figure A.4: Sundberg et al.’s results for the 10th image of the Hamburg taxi sequence, including $\delta$, $f$, region map and distance assignment to regions.
Figure A.5: Sundberg et al.’s results calculated on 10th image of the trees sequence, including $\delta$, $f$, region map and distance assignment to regions.
The next set of figures show the computed flow for the 1st and 2nd models methods. Figures A.6 and A.7 show the optical flow for 1st and 2nd order optical flow for the 6th image of hand sequence, for both uncolored and colored optical flow. Middle and lower flows are for the 1st order optical flow overlaid on the original image and the region map colour by the color of the region the flow vector are in. Figure A.7 shows the 2nd order optical flow for the 6th image of hand sequence, for both uncolored and colored optical flow. Middle and lower flows: 1st order flows overlaid on the original image and the region map colour by the color of the region the flow vector are in. Figures A.8 and A.9 show the same images for the 10th of the Hamburg taxi sequence while Figures A.10 and A.11 show the corresponding images for the 10th image of the trees sequence. Figures A.12, A.13 and A.14 show the warping errors for the full and occlusion regions, superimposed in the 6th hand image, the 10th Hamburg image and the 10th trees images. Table A.1 summarizes the full warping errors while Tables A.2, A.3 and A.4 summarize in table format the individual regions’ warping error.

<table>
<thead>
<tr>
<th>Image</th>
<th>Brox et al.’s</th>
<th>1st Order</th>
<th>2nd Order</th>
<th>Zero flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand 6th image</td>
<td>0.0690</td>
<td>0.0457</td>
<td>0.0443</td>
<td>0.0706</td>
</tr>
<tr>
<td>Taxi 10th image</td>
<td>0.0412</td>
<td>0.0218</td>
<td>0.0216</td>
<td>0.0255</td>
</tr>
<tr>
<td>Trees 10th image</td>
<td>0.0666</td>
<td>0.0440</td>
<td>0.0418</td>
<td>0.0607</td>
</tr>
</tbody>
</table>

Table A.1: Warping error for the 6th image of hand sequence, the 10th image of the Hamburg taxi sequence and the 10th of trees sequence.

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.</th>
<th>1st Order</th>
<th>2nd Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0588</td>
<td>0.0098</td>
<td>0.0109</td>
<td>1.99</td>
</tr>
<tr>
<td>3</td>
<td>0.0344</td>
<td>0.0122</td>
<td>0.0128</td>
<td>0.70</td>
</tr>
<tr>
<td>4</td>
<td>0.0564</td>
<td>0.0152</td>
<td>0.0160</td>
<td>4.35</td>
</tr>
<tr>
<td>5</td>
<td>0.0719</td>
<td>0.0413</td>
<td>0.0390</td>
<td>71.13</td>
</tr>
<tr>
<td>6</td>
<td>0.0516</td>
<td>0.0465</td>
<td>0.0469</td>
<td>20.69</td>
</tr>
</tbody>
</table>

Table A.2: Individual region warping errors calculated for Brox et al. and 1st and 2nd optical flow for 6th image of the hand sequence.

We can see that our 1st and 2nd methods return better results than Brox et al.’s approach. Our methods performs better for the optical flow estimation in every region. That shows boundary
Figure A.6: Top flows: 1st order optical flow for the 6th image of hand sequence, for colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in.

detection does improve the performance of optical flow. However, there is little or no difference between the 1st and 2nd optical flows.
A.2 Optical Flow Overlaid On The Images

We overlay optical flow on the image sequences including rocking horse, tennis and bench, to show the motion.
Figure A.8: Top flows: 1st order optical flow for the 10th image of Hamburg taxi sequence, for colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in.
Figure A.9: Top flows: $2^{nd}$ order optical flow for the $10^{th}$ image of Hamburg taxi sequence, for colored optical flow. Middle and lower flows: $2^{nd}$ order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in.
Figure A.10: Top flows: 1st order optical flow for the 10th image of trees sequence, for both uncolored and colored optical flow. Middle and lower flows: 1st order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in.
Figure A.11: Top flows: 2\textsuperscript{nd} order optical flow for the 10\textsuperscript{th} image of trees sequence, for both uncolored and colored optical flow. Middle and lower flows: 2\textsuperscript{nd} order flow overlaid on the original image and the region map colour by the color of the region the flow vector are in.
Figure A.12: The warping errors calculated on hand sequence. The top images use Brox et al optical flow while the middle images use 1st order flow and the bottom images 2nd order flow.
Figure A.13: The warping errors calculated on Hamburg taxi sequence. The top images use Brox et al optical flow while the middle images use 1st order flow and the bottom images 2nd order flow.
Figure A.14: The warping errors calculated on trees sequence. The top images use Brox et al optical flow while the middle images use 1st order flow and the bottom images 2nd order flow.
## Table A.3: Individual region warping errors calculated for Brox et al. and 1\textsuperscript{st} and 2\textsuperscript{nd} optical flow for 10\textsuperscript{th} image of the Hamburg taxi sequence.

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.</th>
<th>1\textsuperscript{st} Order</th>
<th>2\textsuperscript{nd} Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0308</td>
<td>0.0185</td>
<td>0.0183</td>
<td>93.17</td>
</tr>
<tr>
<td>3</td>
<td>0.0758</td>
<td>0.0259</td>
<td>0.0245</td>
<td>2.41</td>
</tr>
<tr>
<td>4</td>
<td>0.1153</td>
<td>0.0476</td>
<td>0.0497</td>
<td>0.95</td>
</tr>
<tr>
<td>5</td>
<td>0.1319</td>
<td>0.0510</td>
<td>0.0484</td>
<td>1.87</td>
</tr>
<tr>
<td>6</td>
<td>0.1112</td>
<td>0.0450</td>
<td>0.0466</td>
<td>0.69</td>
</tr>
</tbody>
</table>

## Table A.4: Individual region warping errors calculated for Brox et al. and 1\textsuperscript{st} and 2\textsuperscript{nd} optical flow for 10\textsuperscript{th} image of the trees sequence.

<table>
<thead>
<tr>
<th>Region Number</th>
<th>Brox et al.</th>
<th>1\textsuperscript{st} Order</th>
<th>2\textsuperscript{nd} Order</th>
<th>Pixels %</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.0513</td>
<td>0.0313</td>
<td>0.0315</td>
<td>4.45</td>
</tr>
<tr>
<td>3</td>
<td>0.0633</td>
<td>0.0410</td>
<td>0.0377</td>
<td>70.97</td>
</tr>
<tr>
<td>4</td>
<td>0.0729</td>
<td>0.0475</td>
<td>0.0470</td>
<td>19.74</td>
</tr>
<tr>
<td>5</td>
<td>0.1085</td>
<td>0.0867</td>
<td>0.0880</td>
<td>1.23</td>
</tr>
<tr>
<td>6</td>
<td>0.1014</td>
<td>0.0580</td>
<td>0.0573</td>
<td>1.09</td>
</tr>
<tr>
<td>7</td>
<td>0.0571</td>
<td>0.0355</td>
<td>0.0360</td>
<td>0.68</td>
</tr>
</tbody>
</table>
Figure A.15: 1st order method’s optical flow overlaid on the bench images from 006 to 013.
Figure A.16: 1\textsuperscript{st} order method’s optical flow overlaid on the rocking horse images.
Figure A.17: 1\textsuperscript{st} order method’s optical flow overlaid on the tennis images from 740 to 747.
Figure A.18: 2\textsuperscript{nd} order method’s optical flow overlaid on the bench images from 006 to 013.
Figure A.19: 2\textsuperscript{nd} order method’s optical flow overlaid on the rocking horse images.
Figure A.20: 2\textsuperscript{nd} order method’s optical flow overlaid on the tennis images from 740 to 747.
Bibliography


Curriculum Vitae

Name: Hua Meng

Post-Secondary Education and Degrees:
M.Sc. Computer Science
The University of Western Ontario
London, ON
2014 - 2016

B.E. Computer Science
Northeast Dianli University
Jilin, Jilin, P.R.China
2007 - 2011

Related Work Experience:
Research Assistant
The University of Western Ontario
London, ON
2014 - 2016

Teaching Assistant
The University of Western Ontario
London, ON
2014 - 2016