University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Information Technology - Automation and Signal Processing	Spring semester, 2015 Open
Writer: Anne-Marthe Hovden	…… (Writer's signature)
Faculty supervisor: Ivar Austvoll	
Thesis title: Removing Outliers From The Lucas-Kanade Method With A Weighted Median Filter	
Credits (ECTS): 30	
Key words: Optical Flow, Lucas-Kanade, Outliers, Median Filtering, Weights, Error Reduction	Pages: 57 + appendix: 4 pages + Zip Stavanger, 15th of June Date/year

Front page for master thesis
Faculty of Science and Technology
Decision made by the Dean October 30th 2009

Abstract

The definition of optical flow is stated as a brightness pattern of apparent motion of objects, through surfaces and edges in a visual scene. This technique is used in motion detection and segmentation, video compression and robot navigation.

The Lucas-Kanade method uses information from the image structure to compose a gradient based solution to estimate velocities, also known as movement of X- and Y-direction in a scene. The goal is to obtain an accurate pixel motion from an image sequence

The objective of this thesis is to implement a post processing step with a weighted median filter to a well known optical flow method; the Lucas-Kanade. The purpose is to use the weighted median filter to remove outliers, vectors that are lost due to illumination changes and partial occlusions.

The median filter will replace velocities that are under represented in neighbourhoods. A moving object will have corners not just edges, and these vectors have to be preserved. A weighted median filter is introduced to ensure that the under represented vectors is preserved. Error is measured through angular and endpoint error, describing accuracy of the vector field.

The iterative and hierarchical LK method have been studied. The iterative estimation struggles less with single error. Because of this the weighted median filter did not improve the iterative LK-method. The hierarchical estimation is improved by the weighted median and reduced the average error of both angular and endpoint error.

Acknowledgements

This master thesis was written as a completion of the study, Information Technology-Automation and Signal Processing at the University of Stavanger. I would like to thank my supervisor Dr. Ivar Austvoll for his advice. Thank you, Jarle Urdal for the support through the semester. I love you darling!

Contents

1. Introduction

The concept of Optical Flow was introduced by psychologist J.J.Gibson to describe the visual perception of action in an environment [19]. Later this became a term in machine vision to describe local image motion, and the vector field illustrating movement of pixels.

The real challenge of optical flow consists of the trade off between the time consuming mathematical calculations versus the more effective, but less accurate methods [5].

D.Sun, S.Roth and M.J.Black [35] [36] published an alteration of the original Horn-Shunck (HS) method, optimizing it by adding a weighted median filter to the computation. The result was a significant improvement, but at the expense of its efficiency. A simplified version of this altered method showed impressive improvement, with respect to the time. Can this alteration be replicated with similar improvements to other methods?

The objective of this thesis is implementing a post processing step with a weighted median filter to another original method, Lucas and Kanade (LK) [8]. How does the median filter affect the accuracy of the method? How is the angular and endpoint error affected?

The objective will be to test the method on Dimetron, Hydrangea and RubberWhale image sequences from the Middlebury Benchmark [5]. Ground truth from the image sequences will provide an angular and endpoint measurement of the accuracy.

The weight function will be tested against the iterative LK-method. Outliers are more of a problem in the coarse-to fine LK-method. The goal of this thesis is to reduce the outlier source of error.

2. Theory

2.1 Optical Flow

The definition of Optical flow is stated as a brightness pattern of apparent motion to the object, through surfaces and edges in a visual scene [32].

This pattern is represented by a vector for each pixel in an image sequence, tracking the motion of pixels and creating a vectorfield representing the motion scene. A single vector represent a pixel that corresponds to the two dimensional projection of the motion.

This technique is used in motion detection and segmentation, frame interpolation, video compression, robot navigation and three dimensional scene reconstruction.

Horn and Schunck [21] introduced in 1981 a pioneering method of computing optical flow. Temporal changes of gray value structures contains three dimensional motion between the sensor and the environment, as well as the spatial structure of this environment. The idea was that extracting this information would provide a solution to detect motion, track objects, correct for camera jitter, align images etc [28].

The quadratic formulation of Gaussian statistic did not account for reflection, occlusion, motion boundaries and other form of noise causing outliers. Black and Anandan [9] presented a robust estimation to deal with outliers first in 1996. [37]

Lucas and Kanade published also in 1981 their method of computing optical flow, taking a different approach to the problem than Horn and Shunck. They stated that the use of spatial intensity gradient from the image is used to find a good match through Newton-Raphson iteration [8].

Since 1981 new methods have been developed, however HS and LK still remain popular with endless extension and modifications. Barron, Fleet and Beauchemin [7] analysed a number of techniques emphasizing the accuracy and density of measurement. Christmas [11] introduced filtering for the computation of gradient based optical flow [32].

Estimating optical flow is done by numerous methods, however they all share a common procedure.[6]

1 Measure the spatial (∇I) and temporal (I_t) intensity derivatives.

2 Integrate normal velocities into full velocities.

2.2 The Motion Constraint Equation

Estimation of optical flow is based on two principles, brightness constancy and spatial smoothness [37]. These are both methods of solving the motion constraint equation (2.4) with 2 unknowns.

2.2.1 Brightness Constancy

Brightness constancy occurs in surfaces, which have no change in the image intensities over time. This leads to small regions with constant image intensity, and the optical flow equation (2.1) originate from this concept.

Pixels neighbouring a surface should have nearly the same motion as the surface. This phenomenon is called Spatial Smoothness. To track pixels, look for nearby pixel of similar image intensities.

Figure 2.1 is an illustration of the brightness constancy concept. The two images are taken from the same position and capture the motion of an object. While the position change, the image intensity of the object remain. The motion can then be defined by equalizing the intensity equations.

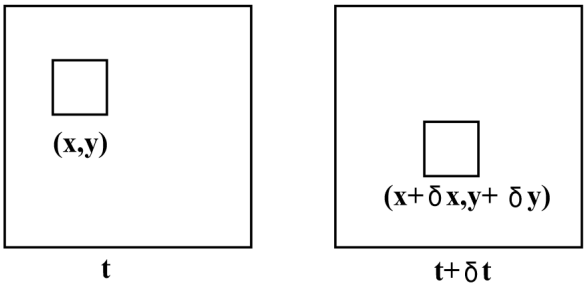


Figure 2.1: Images of a moving object, illustrating equation (2.1). [6]

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{2.1}$$

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + Higher\ Order\ Term \tag{2.2}$$

Equation (2.1) is expanded by a first order Taylor series expansion to ultimately obtain equation (2.4). The higher order term can be ignored, due to small insignificant size.

$$I(x, y, t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \tag{2.3}$$

$$\begin{aligned}
\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t &= 0 \\
\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} &= 0 \\
\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} &= 0 \\
I_x v_x + I_y v_y + I_t &= 0 \\
(I_x, I_y) \cdot (v_x, v_y) &= -I_t
\end{aligned}$$

$$\nabla I \cdot \vec{v} = -I_t \tag{2.4}$$

The image velocities $\vec{v} = (v_x, v_y)$, are the estimated optical flow. $\nabla I = (I_x, I_y)$ and I_t are the image intensity derivatives, and these are computed from a gray scaled image.

The motion constraint equation (2.4) is limited by 2 unknowns and 1 equation. The solution is obtaining as many equations as there are unknowns. Lucas-Kanade uses a gradient based approach, see chapter 2.3

2.2.2 The Smoothness Condition

Instead of using the brightness constancy, a smoothness condition can be applied [17]. The conditions are then

A) Spatial Smoothness

B) Temporal Smoothness

C) By their combination

Spatial smoothness introduces an additional condition to minimize the square of the magnitude in the gradient. Error is minimized to find the suitable value for the velocities

(v_x, v_y), and these are used to calculate the gradient constraint equation:

$$I_x^2 v_x + I_x I_y v_y = \alpha^2 \nabla^2 v_x - I_x I_t$$
$$I_x I_y v_x + I_x^2 v_y = \alpha^2 \nabla^2 v_y - I_y I_t$$

The calculations are computational complex, and should be solved by an iterative method. The HS method use spatial smoothness to solve the gradient constraint equation.

The temporal optimization method, smoothness is expressed as $\left(\frac{\partial v_x}{\partial t}\right)^2$ and $\left(\frac{\partial v_y}{\partial t}\right)^2$.

2.3 Gradient based estimation

The motion constraint equation have two solutions, a local and a global.

The local solution computes the intensity derivatives for a small window or area in an image, which provides a robust computation under noise, but not a dense flow field.

The global solution attempt to minimize a global energy function. This method provides a dense flow field but is more sensitive to noise.

Bruhn, Weickert and Schnörr [10] have explored if the global and local solutions can be combined to achieve a technique that is robust under noise and produce 100% density fields. This would make the post processing step, where sparse data is interpolated, obsolete.

The LK-method, the focus of this thesis, uses an adaptive window. The HS-method on the other hand optimizes the flow field using brightness constancy and flow smoothness [30].

The motion constraint equation have several solutions, but the most common is increasing the number of equations. Some requirements have to be evaluated [17].

A) The Spatial Local Optimization Method, assumes constant velocity over each spatial neighbourhood. This method is among others used by Kearney, Thompson and Boley [24] and Lucas and Kanade [8]

B) The Temporal Local Optimization Method. Assumes constant velocity over temporal neighbourhoods. This method was used by Kearney, Thompson and Boley [24] and Nomura, Miike and Koga [29].

C) The Multispectral Constraint Method, uses three channels for each pixel. Markandey and Flinchbaugh [27] and Woodham [40] used this method.

D) The Second Order Derivative Method of each pixel. This method is used by among other Bainbridge-Smith and Lane [3], Nagel [28], Tretiak and Pastor [38] or Uras, Girosi et al [39].

E) A combination of the solutions above.

The focus of this thesis is to use the gradient based Lucas-Kanade (LK) method to compute the velocity field. The Spatial Local Optimization method in the LK-method, use information about regions. A region of pixels is considered to have the same velocity. The motion constraint equation is then solved by obtaining information in a small spatial neighbourhood.

Two equations are sufficient to obtain a unique solution for $\vec{v} = v_x, v_y$. More than two equations reduce the effect of error in the observed equations from figure 2.2. The spatial neighbourhood Ω, is equal to $n \times n$ pixels, producing n^2 equations.[17]

$$\nabla I = \begin{bmatrix} I_1 x & I_1 y \\ .. & .. \\ I_i x & I_i y \\ .. & .. \\ I_{n^2} x & I_{n^2} y \end{bmatrix}, b = - \begin{bmatrix} I_1 t \\ .. \\ I_i t \\ .. \\ I_{n^2} t \end{bmatrix}, \vec{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

The gradient constraint equation can then be solved, as illustrated from the figure below.

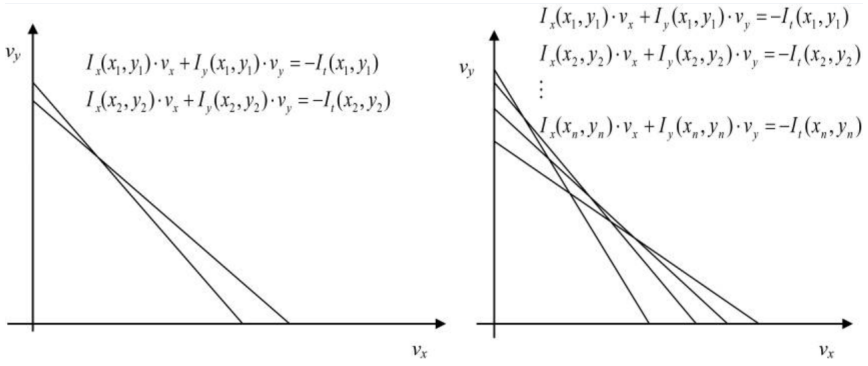


Figure 2.2: Computed constraint line [32]

2.3.1 Aperture problem

The 2 unknowns, 1 equation problem with the motion constraint equation (2.4) is a consequence of the aperture problem.

The aperture problem is the limitation of insufficient local image intensity structures to measure full image velocity. The component normal to the local intensity structure is usually available. [6] [32]

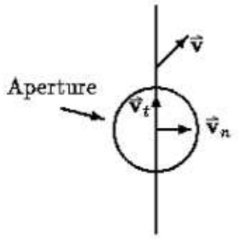


Figure 2.3: The aperture problem [6]

The figure (2.3) shows a moving line through circular aperture. It is impossible to recover full image velocity, however the image velocity normal to the line is available. The problem is finding an additional constraint that yields a second equations with the same unknown $\vec{v}$.

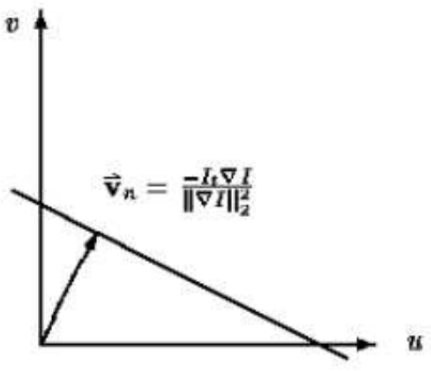


Figure 2.4: Finding the additional constraint with the same unknown as $\vec{v}$ [6]

The motion constraint yields a line in $\vec{v}$ space.

$$v_n = \frac{-I_t}{||\nabla I||_2}$$
$$\hat{n} = \frac{(I_x, I_y)}{||\nabla I||_2}$$

$$\vec{v}_n = v_n \hat{n} = \frac{-I_t(I_x, I_y)}{||\nabla I||_2^2} \tag{2.5}$$

The LK-method allows estimation of the normal velocity via a least square calculation, and the motion constraint equation can be re-written :

$$\vec{v} \cdot \hat{n} = v_n \tag{2.6}$$

The eigenvalues of the Hessian matrix $[A^T A]$ are of interest for both the Harris Corner detection and to test if Aperture is a problem in the local window estimation. These eigenvalues are tested against a defined threshold, often 1. Then certain precautions are defined from this results. The figure (2.5) illustrates the information obtained from this matrix.

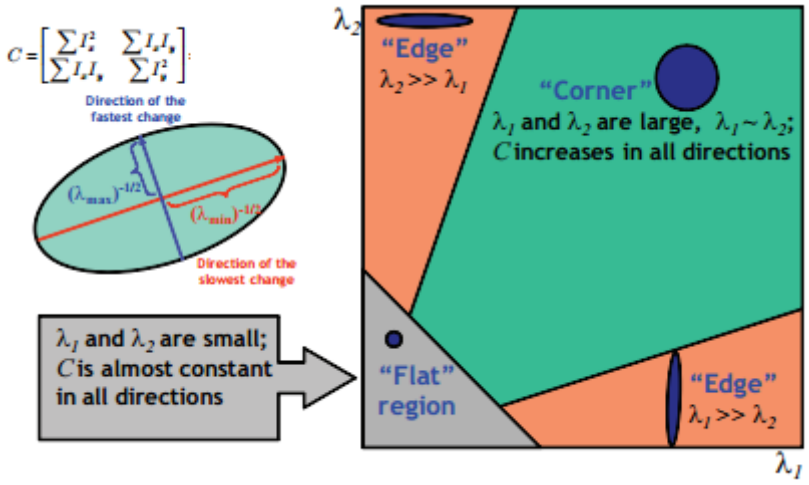


Figure 2.5: The hessian marix and the result of its eigenvalues [16]

Given $\lambda_2 \geq \lambda_1 \geq 0$, the orhogonal eigenvectors $\hat{e}_1$ and $\hat{e}_2$ are obtained.

If the smallest eigenvalue $\lambda_1 \geq \tau_D$, thus bigger than 1, then Aperture is still not a problem in the local window. The velocity is calculated from the usual equation (2.20).

However Aperture is a problem if the biggest eigenvalue is larger than set threshold, $\lambda_2 \geq \tau_D$. The minimum eigenvalue is less than the threshold, $\lambda_1 < \tau_D$. The least square

calculation can be estimated by projecting $\vec{v}$ in the direction of the large eigenvalue, as shown in equation.

$$\vec{v}_n = (\vec{v} \cdot \hat{e}_2)\hat{e}_2 \tag{2.7}$$

2.4 The Image Intensity Derivatives

The spatial I_x, I_y and temporal I_t intensity derivatives are obtained from the image by a convolution between the grayscaled, gaussian filtered image and the filter m_x, m_y and m_t.

$$\begin{aligned} I_x &= m_x * (I_1 + I_2) \quad m_x = \tfrac{1}{4}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \\ I_y &= m_y * (I_1 + I_2) \quad m_y = \tfrac{1}{4}\begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix} \\ I_t &= m_t * (I_2 - I_1) \quad\quad m_t = \tfrac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \end{aligned} \tag{2.8}$$

Intensity changes occurs when there are changes in the image scene. Changes can be in depth, surface color and texture. It can also be discontinuities in orientation and surface color and texture. These occurrences all comes from geometric events in the scene.

Non-geometric events can be specularities, shadows or inter-reflections.

The intensity derivatives are also known as a gradient. With this name magnitude and direction are important followers. The magnitude of the gradient inform of the change of the image, while the direction tell of the direction of which the image changes most rapid.

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{2.9}$$

To differentiate, derivative filters are used. [-1 1] and [-1 0 1] are two well known filters for this purpose. By using the former, the directional filters can be expressed by:

$$m_x = \begin{pmatrix} -1 & 1 \end{pmatrix}, m_y = \begin{pmatrix} -1 & 1 \end{pmatrix}^T, m_t = \begin{pmatrix} 1 & 1 \end{pmatrix} \tag{2.10}$$

The equations can then be expressed as:

$$\frac{\partial I}{\partial x} \approx I(x-1, y) - I(x, y) \tag{2.11}$$

$$\frac{\partial I}{\partial y} \approx I(x, y-1) - I(x, y) \tag{2.12}$$

This latter filter look at changes over a wider region, $2\Delta x$ and $2\Delta y$, and can be expressed by:

$$m_x = \frac{1}{2}\begin{pmatrix}-1 & 0 & 1\end{pmatrix}, m_y = \frac{1}{2}\begin{pmatrix}-1 & 0 & 1\end{pmatrix}^T, m_t = \frac{1}{2}\begin{pmatrix}1 & 1 & 1\end{pmatrix} \tag{2.13}$$

The filter derivatives over this area:

$$\frac{\partial I}{\partial x} \approx \frac{I(x+1, y) - I(x-1, y)}{2} \tag{2.14}$$

$$\frac{\partial I}{\partial y} \approx \frac{I(x, y+1) - I(x, y-1)}{2} \tag{2.15}$$

2.5 The Lucas-Kanade Method

The LK method is a least square (LS) minimization of the 2D gradient constraint equation (2.4) with a gaussian window $g(x, y)$. The spatial intensity gradient information is used to direct the search for the position that yields the best match. [32]

One of the conditions to the LK-method is brightness constancy. Spatial and temporal intensity derivatives are obtained from an gray scaled intensity image. The LK-method uses the spatial local optimization method, which assume constant velocities over a spatial neighbourhood $\Omega = n \times n$. Gaining the gradient constraint equations, the velocities can be calculated. A Least Square estimation is introduced to obtain the most accurate estimate. Equation (2.16) is a least square calculation of the gradient constraint multiplied with a gaussian window to minimize error [34].

$$E_{LK}(v_x, v_y) = \sum_x g(x, y)\big(I_x v_x + I_y v_y + I_t\big)^2 \tag{2.16}$$

Where $g(\vec{x})$ is a gaussian function that determines the support of the centred estimator. The partial derivative is calculated from the minimized error estimate, obtaining equation (2.17).

$$\begin{bmatrix} \frac{\partial E(v_x, v_y)}{\partial v_x} \\ \frac{\partial E(v_x, v_y)}{\partial v_y} \end{bmatrix} = \begin{bmatrix} \sum_x g(x, y)\big(v_x I_x^2 + v_y I_x I_y + I_x I_t\big) = 0 \\ \sum_x g(x, y)\big(v_y I_y^2 + v_y I_x I_y + I_y I_t\big) = 0 \end{bmatrix} \tag{2.17}$$

The partial derivative is the result of the smoothing constraint, where the additional condition is to minimize the square of the magnitude in the gradient. With this in mind, the gradient constraint equation can be rewritten as:

$$\begin{bmatrix} \sum g(x,y)I_x^2 & \sum g(x,y)I_xI_y \\ \sum g(x,y)I_xI_y & \sum g(x,y)I_y^2 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = - \begin{bmatrix} \sum g(x,y)I_xI_t \\ \sum g(x,y)I_yI_t \end{bmatrix} \tag{2.18}$$

$$A^TA \cdot \vec{v} = A^T\vec{b} \tag{2.19}$$

$$\vec{v} = [A^TA]^{-1}A^T\vec{b} \tag{2.20}$$

$$A = [\nabla I(x_1, y_1), ..., \nabla I(x_N, y_N)]$$
$$\vec{b} = -(I_t(x_1, y_1), ..., I_t(x_N, y_N))$$

2.5.1 Iterative LucasKanade

Solving the iterative LK algorithm is done by repeating the process until the desired result is obtained.

Iterative LS computation is using the metrics in the local window to find the velocities v_x and v_y. From the gradient in the local window, the Hessian matrix H is found. The Hessian matrix is defined as $H = A^TA$, and contains information of the local window region.

The LK-method estimates the motion of objects from a sequence of images. Velocity of the image is generated from the transformation of one image into the next, like figure 2.6 illustrates.

LK is a local optimization problem that can not preform properly if the movement are too large. The gradient information is obtained from neighbouring pixels and the real motion can not extend beyond the local region. Local neighbourhoods are finite for the LS approach and this limits the estimation of velocities in large movements. It is common to use a pyramidal approach to this problem [32].

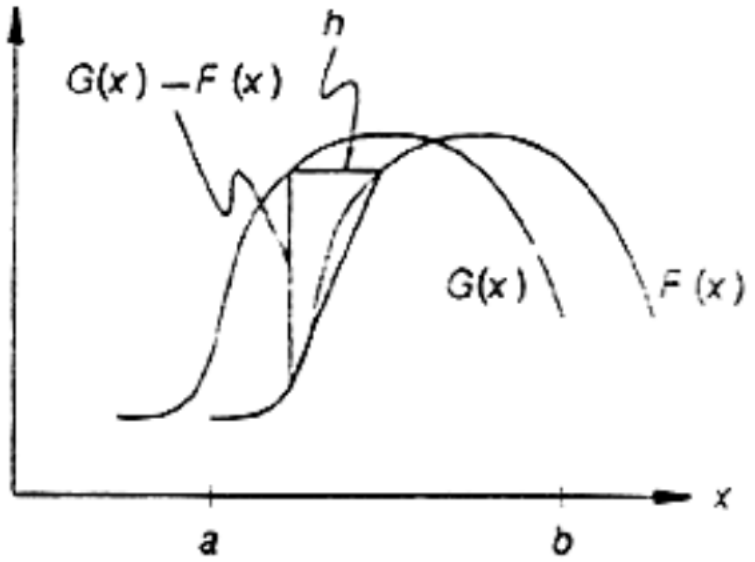


Figure 2.6: The velocity, illustrating the transformation from one image to the next [8].

2.5.2 Coarse-to-Fine Lucas-Kanade

In the coarse to fine approach the input images are resized to a lower resolution. First filtering with a LP filter, then subsampled by the coarse-to-fine approach.

The image gradients are first computed on the top level of the pyramid, then upsampled and used to initialize the estimate at the next level, illustrated in figure 2.7.

In the image pyramid a generic image im1 of size $n_x \times n_y$ where the $im1^0$ is the zero level of the image. This is the highest resolution image. The pyramid representation is then built in a recursive fashion from im^0 to im^1, and from im^1 to im^2.

The input images are resized to a lower resolution, first by filtering with a low pass filter and then subsampled by a factor of 2, technique called coarse-to-fine approach, as shown in Figure 2.7. The computation of the optical flow is started with the lowest resolution images, at the highest pyramidal level. The result is then passed on to the higher resolution level as an initial estimate. Running the algorithm on higher resolutions will cause higher accuracy of the flow field.

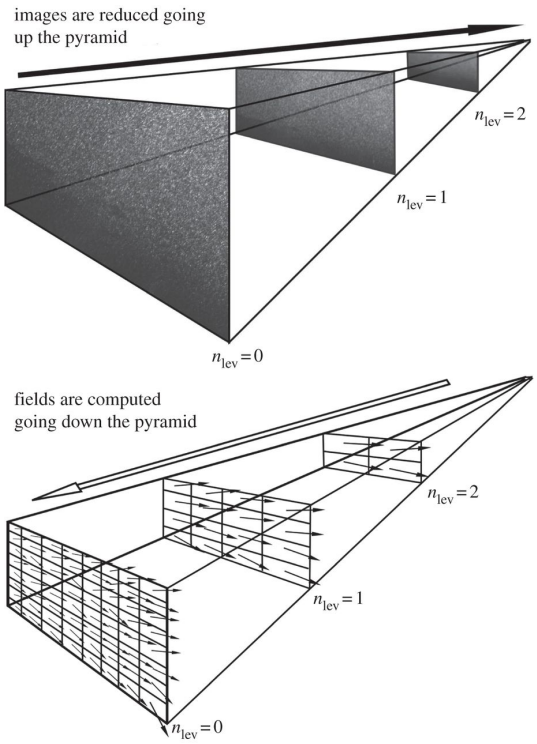


Figure 2.7: The coarse-to fine implementation[18].

2.6 The Goal

When D.Sun, S.Roth and M.J.Black published their improvement on the HS-method [35], it ranked as number 1 on the Middlebury Benchmark evalution [1]. They published another study in 2013 [36], on the results gained. The method was called non-local classic, and at june 2015 it ranked as 33.3 on average angular error and 31.4 on average endpoint error.

The non-local classic method was an implementation of the discovered improvement a weighted median filter had on the method.

The Harris corner detector detects corners, edges and flat regions of an image. The goal is to use this detector to implement weights to corners. Edges are more prone to error, because of the aperture problem and should have zero weights.

Though their method is not still ranked at the top of the Middlebury evaluation, it is still interesting to see what made these improvements. Can these improved results can be reproduces with another established method.

The goal of this thesis is to see if the weighted median removes outliers and reduces error on the LK-method.

2.6.1 The improvement of HS 2013

The goal of D.Sun, S.Roth and M.J.Black publication was to see what made the leading optical flow estimation successful [35] [36]. Results showed that applying a median filter to intermediate flow values during incremental estimation and warping made the most significant improvement. Both improved the accuracy of the recovered flow fields and increased the energy of the objective function. By using Li and Oshers observations [26] on the L1 energy function, a non local term was added. They stated that the new term integrated information over large spatial smoothness. They also incorporated a weighted median filter to prevent over smoothing across boundaries.

At the point of the first publication, it ranked 1st in Angular and Endpoint error on the Middlebury evaluation.

The median filter used is a 5×5 window, that performed better than both 3×3 and 7×7. The 5×5 filter was applied on the window after every warping iteration.

2.7 Improving the LK-method

The LK-method is not known to produce accurate vector fields. Vectors representing corners and edges are often under represented in a local areas, and overfiltration is a real concern. These vector values are often lost during compilation filtering. By weighting important pixels in the image, this can be prevented.

The Harris corner detector determine flat regions, edges and corners from eigenvalues of the Hessian matrix $H = A^T A$. Information about image structure distribute weights to corners, which is more likely to contain a moving object.

The median filter have a reputation of being more sensitive to corners and edges than a linear filter. This undocumented hypothesis is put to the test in this thesis.

2.7.1 Harris corner detector

Optical flow represent movement in an image sequence. The moving part of the image is an object, defined by structures, edges and corners.

The Hessian matrix is defined as

$$H = \sum (\nabla I)(\nabla I)^T \approx \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = A^T A \tag{2.21}$$

This matrix is a vital part of the LK method and is already defined in equation (2.20).

The corner detector is used do distribute weights to corners to define higher weights of pixels. Edges provide more pixels with weight, emphasizing more than the moving object. Corners are more precise, because more objects have edges, but not all have corners. The moving object will have corners. The Harris corner detector is given by:

$$det(H) - k(\frac{trace(H)}{2})^2 \tag{2.22}$$

Where $Det(H)$ is the determinant of the Hessian matrix H, and trace is the sum of the eigenvalues of H. K is some constant.

The Harris corner detector looks for the local maxima of equation (2.22). The results are then tested against a threshold, which check whether the eigenvalues is larger than the square of the average.

Large local maxima values imply big eigenvalues. The eigenvalues are big at corners, edges score some what lower on the scale, while flat regions should produce small values[20].

These values are normalized to a scale from 0 to 1, with the purpose of weighting the median filter. By this definition, if a pixel is located in a neighbourhood with a corner, these should be weighted. Edges return negative values from the Harris corner detector, and the scaling sets these values to zero, and providing no weights to edges. Flat regions should score low on the normalized scale.

2.7.2 Outliers

Outliers are lost pixels due to illumination changes or partial occlusion. Intensity changes illustrates a moving object, and changes in illumination in a static object illustrate a false movement. These point should be rejected in the process. [41] There are several methods for outliers rejection, but the RANSAC method is the most common in optical flow.

The RANSAC method randomly select a subset of the data and determine the outliers from dataset. The process is repeated trough a prescribed number of iteration. the method is capable of interpreting data containing a significant error and is suitable for automated image analysis [14].

Rejecting outliers creates holes in the flow field, and the denisty if no longer classified as 100%. However large inaccurate data is useless, and a minor decrease in the density is preferred [2].

By using a threshold, the density can be traded against accuracy. The measurement is defined by the equation:

$$Density\ Measure = \frac{Number\ Of\ Computed\ Velocity\ Vectors}{Total\ Number\ of\ Velocity\ Vectors} \times 100\% \tag{2.23}$$

2.7.3 Why Median filtration

Instead of using the RANSAC method, this thesis study how a weighted median filter affect outliers. In theory, outliers will be replaced with the median neighbouring pixels, as the pixel is under represented in the neighbourhood.

Median filtering finds the median over a sliding window of fixed size. In an iterative case, median filtering would achieve noise reduction with smaller bias near edges.

Median filters, applied to an image, filter the pixel based on the median filter mask. To a vector field, the decision on length and angle have to be made. Whether to filter on length, angle or both.

2.8 AAE and AEE

The estimated performance of the vector field is measured through Average Angular Error (AAE) and Average Endpoint Error (AEE).

Ground truth is used to estimate the accuracy of the method and is defined as the true motion field. With this measurement available, it becomes possible to compare the estimates to the ground truth. Image sequences from the middlebury benchmark [1] provides ground truth.

AAE is a an accuracy determinant of the angle of the vector, defined by the equation (2.24).

$$AAE = \cos^{-1}(\hat{c} \cdot \hat{e}) \tag{2.24}$$

Where $\hat{c}$ is the ground truth and $\hat{e}$ the vector estimates.

To get an accurate estimate on the length of the vector, AEE is defined by equation (2.25).

$$AEE = \sqrt{(v_x - v_{GTx})^2 + (v_y - v_{GTy})^2} \tag{2.25}$$

The need for ground truth limits the available test sequences. From the Middlebury benchmark, Dimetron, Rubber Whale, Hydrangea from figure 2.8 are all image sequences

with public available ground truth. These sequences are real images with hidden texture.

Figure 2.8: The test sequences: Dimetron, Rubber Whale and Hydrangea.

3. Implementation

The implementation of the Lucas Kanade method was done using Matlab R2012b.

Deqing Sun, from the Department of Computer Science at Brown University published his implementation of weighted median filtration on the HS method, together with his article [35]. The source code from the article is available, and made it possible to review both his implementation and results. Several of his functions have been to great use during the implementation in this thesis, among others compare the results to the ground truth and a measurement of angular error.

Stefan Roth, from the Department of Computer Science at TU Darmstadt published a function called plotflow(), effectively implementing the vectorfield by arrows. This made it possible to review the flow field of the estimates, and discuss regions of error. He also published several functions making it easier to review the results. E.g. *flow to Color*, a more precise method of reviewing challenging areas of the estimates.

The University of Central Florida and author Sohaib Khan published lab material on the Lucas-Kanade method [25], consisting of a basic LK-computation of the equation 2.20.

The implementation of Lucas-Kanade by Khan is used as a base for this thesis. Some adjustments was however made to the original code. The derivative filter was changed to a more suitable error reducing filter. To smooth the image and reduce computational error, a pre processing step using a gaussian filter was added. These conditions had to be implemented before the real goal of the thesis started; minimizing the outlier error by using a weighted median filter.

3.1 Iterative Lucas-Kanade

The iterative approach of implementing Lucas-Kanade is based on iterating through every element of the image, calculating the desired flow using a defined window. Algorithm 1

shows a simple implementation of equation (2.20), with an added Harris detetor to compute a weight matrix. The additional weight computation will be described in section 3.3.

Algorithm 1: LucasKanade

Input: image1 $im1$, image2 $im2$, $windowSize$
Output: velocity u, velocity v, weight matrix ω

Initialization
$halfWindow = \lfloor windowSize/2 \rfloor$
$[fx, fy, ft] = ComputeDerivatives(im1, im2)$

for *element i in row* **do**
  for *element j in column* **do**
    $curFx$ = window centered around element in matrix Fx
    $curFy$ = window centered around element in matrix Fy
    $curFt$ = window centered around element in matrix Ft

    $curFx = curFx^t$
    $curFy = curFy^t$
    $curFt = curFt^t$

    $curFx = vec(curFx)$
    $curFy = vec(curFy)$
    $curFt = -vec(curFt)$

    $A = [curFx, curFy]$
    $H = A'A$
    $U = H^+A'curFt$

    $w(i,j) = |H| - Constant * ((tr(H))/(2))^2$

    $u(i,j) = U(1)$
    $v(i,j) = U(2)$
  end
end
Replace all NaN in u and v with 0

The function *ComputeDerivatives* used in the algorithm computes the needed derivatives for the given images. *Fx* describes the intensity change in the horizontal direction for both images, and *fy* the intensity change in the vertical direction. While *ft* describes the difference between the two images.

$$fx = im1 * filter_x + im2 * filter_x$$
$$fy = im1 * filter_y + im2 * filter_y$$
$$ft = im1 * filter_t - im2 * filter_t$$

This method iterates through the image pixel elements. Given a defined pixel in the image, a corresponding window centered around the element is found for all three derivatives.

All three windows are transposed and vectorized. The intensity derivate *(A)* are defined as the matrix consisting of change in horizontal and veritcal direction. A Hessian matrix is then created by using the intensity derivates. Flow is finally computed by taking the pseudoinverse of the Hessian matrix multiplied with the transpose of the intensity derivative and time derivative.

3.2 Hierarchical Lucas-Kanade

The Hierarchical approach of implementing Lucas-Kanade uses the known image pyramid. Flow is first found by studying a reduced version of the original image, this flow is then used as an initial value when calculating the flow for the next step in the pyramid. A simple implementation of the method is shown in algorithm 2.

Algorithm 2: HierarchicalLK

Input: image $im1$, image $im2$, pyramid levels $numLevels$, smoothing window size $windowSize$, $iterations$

Output: length matrix u, angle matrix v, $cert$

Build pyramid
Set $pyramid1 = im1$ and $pyramid2 = im2$
for $i = 2$ *to* $numLevels$ **do**
  Reduce $im1$ and $im2$ by LP-filter and subsample the image by a factor of 2
  Insert $im1$ into a new layer in $pyramid1$ and $im2$ into a new layer in $pyramid2$
end

Base level computation
Set $baseIm1$ and $baseIm2$ as the smallest image in the corresponding pyramids
[u,v] = LucasKanade($baseIm1$, $baseIm2$, $windowSize$)
for $i < iterations$ **do**
  [u, v] = LucasKanadeRefined(u, v, $baseIm1$, $baseIm2$);
end

Propagating flow to higher levels
for $i = 2$ *to* $numLevels$ **do**
  Expand u to uEx and v to vEx by scaling the matrixes with a factor of two.
  Multiply uEx and vEx with a factor of two.
  Set $curIm1$ and $curIm2$ as the pyramid images corresponding to the size of uEx and vEx
  [u, v] = LucasKanadeRefined(uEx, vEx, $curIm1$, $curIm2$)
  for $i < iterations$ **do**
    [u, v, $cert$] = LucasKanadeRefined(u, v, $curIm1$, $curIm2$)
  end
end

In addition an initialization step should be implemented as this method has some limitations on inputs. Both images has to be greyscale, have identical size and be dividable by $2^{numLevels-1}$. These requirements can easily be implemented by modifying the input images.

The method is divided into three distinct sections: building pyramid, base level computation and propagating flow to higher levels. The first section creates the image pyramid from a bottom-up approach by lowpass filtering the image and subsampling with a factor of two. The second section calculates the base flow for the smallest image in the pyramid, used for propagating flow in higher levels. The third section propagates the previously calculated flow to the next level in the pyramid by scaling the estimates by a factor of two, and using this flow as an estimate in the next calculation. As the method *LucasKanadeRefined* computes flow given previous estimates, this method is run several times. This is defined by setting the parameter *iterations* in algorithm 2.

3.3 Median Weight filter

Sun, Roth and Black suggest in their thesis [35] implementing a weighted median filter, to allow defining the importance of individual pixels. The challenge is to determine which pixels. A weight function should be used to determine pixels with higher importance, and use this information when filtering the image.

If the computation struggles with single error outliers, adding a weighted median filter should in theory improve the result. The filter will only improve single error velocities, not affecting the global accuracy of the velocity estimation.

This thesis has chosen to compute weights based on the Harris corner detector. By studying the eigenvalues from this detector, we can distinguish corners, edges and flat regious in the image. The resulting matrix of eigenvalues can then be used for creating weights based on the desired property. As the weights are directly based on the eigenvalues from the detector, large values can appear in the weight matrix. To avoid this, we have chosen to normalize the weights in the region $0 - 1$. Where a weight of 1 means the pixel element corresponding to the weight will be used directly, while a 0 means the pixel element corresponding to the weight will be discarded.

The weight was applied as a post processing step, after the velocities were estimated.

3.3.1 Weighted Filter

A simple algorithm of the developed weighted filter is shown in algorithm 3. The method require an already computed weight matrix w. This thesis uses the Harris corner detector as a base for the weight matrix, and the computation for this is done in the modified *LucasKanade*, seen in algorithm 1.

Algorithm 3: Weighted median filter

Input: Matrix A, weight matrix ω, $windowSize$
Output: Weighted matrix B

Initialization
$halfWindow = \lfloor windowSize/2 \rfloor$
Define a sigmoid weight function, s

for *element i in row* **do**
  for *element j in column* **do**
    Extract window centered around element from ω as ω_window
    Find $\omega 01$ by normalizing ω_window to values between zero and one
    Find $\omega 01s$ by applying s to the normalized window
    Find *window* as the Hadamard product between ω_01s and the corresponding window from A
    Set B(i,j) as the median of *window*
  end
end

To allow for non-linear weighting, a sigmoid function is chosen as a weight function. The ideal weight curve is unknown and have to be tested. The sigmoid function is chosen as it can easily represent a step, non-linear functions and linear functions.

The method iterates though the elements in the input matrix. For each element a window centered around the element is extracted from both the weight matrix and the input matrix. The weight window is then normalized in the region $0 - 1$ before applying the sigmoid function to calculate the final weights. The weights are then applied to the input window by calculating the Hadamard product between the two matrices. The resulting pixel element is then found as the median of the weighted window.

Weighting should be done by assigning small weights to regions of low importance, like flat regions. The point of the weighting is to ensure corners or edges are not over filtrated. These vectors are under represented in a local window and should be distributed weights to ensure that their movement is preserved.

Edges can sometime struggle with the aperture problem, where one velocity can be measured, but there is insufficient image intensities to measure the other. The normal velocity

of the other velocity is the solution to this problem, however this can sometime increase error.

Because of this problem, the edges have been distributed zero weights. The print of the cloth, in the Dimetron sequence from figure have edges, but there is no movement here. Moving objects will however always have corners, and we have because of this chosen to emphasize on this property.

4. Results

The main challenge of implementing the LK-method is its sensitivity to change. Which is why it is important to define a starting point of the computation method.

Common problems with images are large motions, which the iterative LK-method struggles with. The solution is using the hierarchical LK-method, unfortunately does this method also struggle with the large motion of small objects.

Other problems in scenes are the degradation of camera such as sensor noise. Illumination change and atmospheric effects are also problems along with high specularities and transparent materials.

The hidden texture data contains a number of moving shadows and other illumination related effects. These problems are discussed in the following sections.

4.1 Dense flow field

Some velocities lack sufficient information to be estimated and in the ground truth these flow values have been set to an abnormal high number, 1×10^9. This means that there are holes in the flow field and the density is less than a 100%.

To calculate the density, a matrix of the same size as the images is created. If the ground truth velocity is 1×10^9 or higher, the matrix element is defined as 1. If the value is less, the element is defined as 0. To calculate the number of holes in the density field, the sum of the matrix elements are calculated

The process is defined as follows for the Dimetron sequence.
Holes in the flow field = 10 772
Total Number of pixels = 226 592

$$\frac{10772}{226592} \times 100\% = 4.75\% \tag{4.1}$$

Image	Number of Unobtainable	Density
Dimetron	10 772	95.25%
Hydrangea	14 880	93.43%
RubberWhale	3 622	98.40%

Table 4.1: The density of the hidden texture sequences.

4.2 Weight function

The Harris corner detector is used to weight the image pixels in a median filtration. The weight is based on the retrieved information from the eigenvalues of the Hessian matrix given in equation (2.21).

Scaling is done by creating a weighted window using the Harris output values in the window surrounding each pixel element. The weight window is normalized with focus on detected corners, letting these pixels weight extra under the median filtration.

The normalized values are plotted with several curves to test several weight functions. The sigmoid function, a linear curve and a step function distribute weights differently.

The linear weight function was quickly discarded as a solution, when the error increased under this condition. The step and the sigmoid function showed small improvements on the median filter.

The iterative LK-method uses approximately 30 seconds to compute the velocities and plot them into a vector field.

4.2.1 Linear

At first the linear weight function, shown in figure 4.1 was applied. This would apply weights to corners, but also some to the flat regions of the scene.

AAE	AAE Weighted	AEE	AEE weighted
Dimetron Image sequence:			
11.1036	13.3178	0.5300	0.5939
Hydrangea Image sequence:			
11.2121	11.7917	1.1031	1.1291
RubberWhale Image sequence:			
15.8819	15.2370	0.4981	0.4653

Table 4.2: Measured error of a linear weight function.

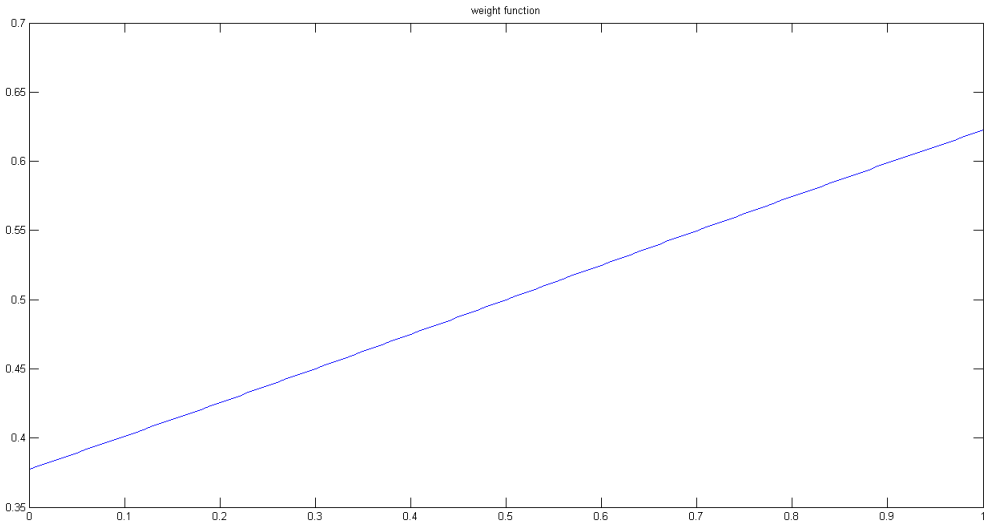


Figure 4.1: The weighted linear function.

The error table 4.2 represent the error of the iterative computation using a linear weight function. This function does not work, because it distribute most weight to corners, but also some weights to flat regions. Only the RubberWhale image sequence is improved by this filter, which is due to the nature of the median filter and the underrepresented outliers.

4.2.2 Sigmoid

Another weighting function taken into account is a step function and a sigmoid curve. Both give large weights to corner, but varies in weights to the flat regions. The step function is only a stricter version of the s-function. It distribute weights to only to corners. Unlike the s-function which might distribute some depending on turning point. Figure 4.2 shows the applied S-function to the median filter.

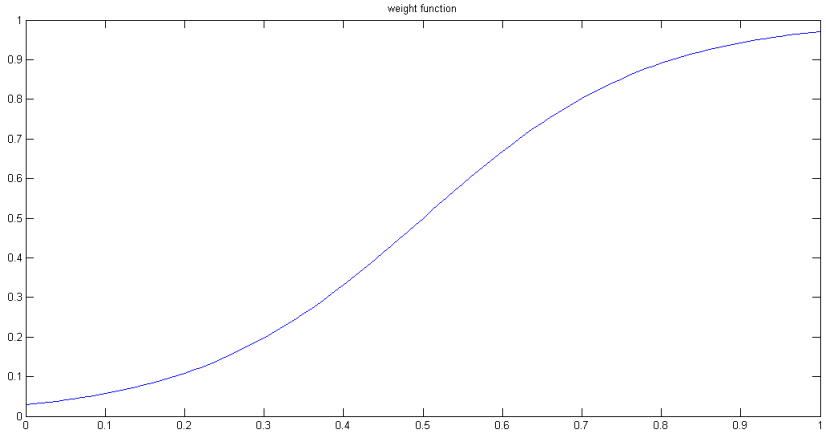


Figure 4.2: The applied S-function to compute the weights of the image.

AAE	AAE weighted	AEE	AEE weighted
Dimetron Image sequence:			
11.1036	11.1986	0.5300	0.5239
Hydrangea Image sequence:			
11.2121	11.4050	1.1291	1.0884
RubberWhale Image sequence:			
15.8819	14.8946	0.49812	0.4553

Table 4.3: Measured error of a weighted S-function.

Figure 4.3 show the computed error using the computed median filter and the applied S-function. The endpoint error is consistently improved, but the angular error is only improved at the hydrangea sequence. The weight function still applies some weight to the flat regions of the scene.

4.2.3 Step

The Harris detector distributes low values for flat regions, and higher for corners. To compute weights, these values are normalized from 0 to 1. This scaling makes the new values for the flat regions minimal.

The chosen turning point is illustrated in figure 4.3.

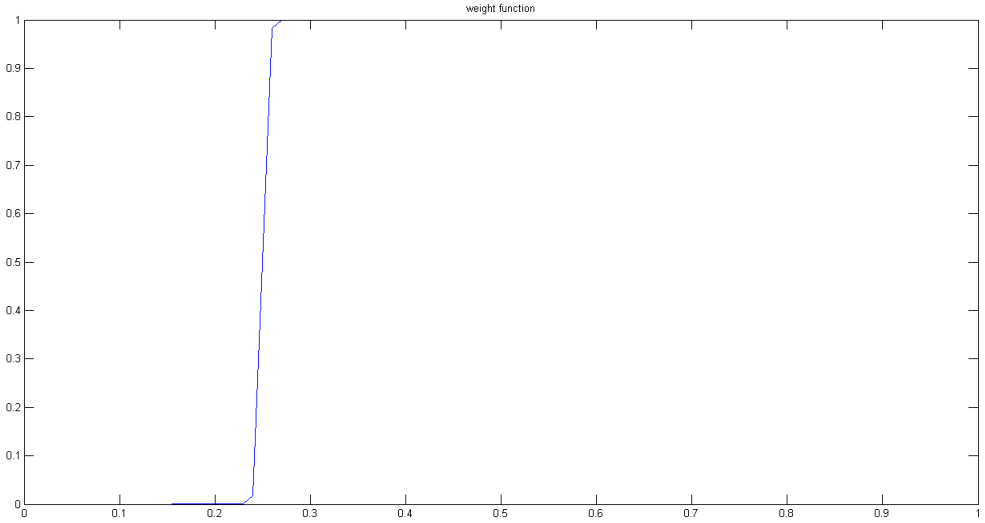


Figure 4.3: The applied step function to compute the weights of the image.

AAE	AAE weighted	AEE	AEE weighted
Dimetron Image sequence:			
11.1036	10.0989	0.5300	0.4899
Hydrangea Image sequence:			
11.2121	11.3818	1.1291	1.0884
RubberWhale Image sequence:			
15.8819	14.5701	0.4981	0.4469

Table 4.4: Measured error of a weighted Step-function.

The median filter works on the endpoint error, however the improvement is small. The angular error is improved in 2 out of 3 image sequences, when applying the weighted median on the iterative LK-method.

The iterative computation does not struggle with anomalies the same way the hierarchy computation does. This weighted median does not seem to have a big impact on the iterative computation.

4.3 Applying the weighted median filter

Information about the vector field is needed when deciding where to apply weights. The U vector represent movement of the X-direction, while the V direction represent movement of the Y-direction.

Logic tells of movement biggest at the U velocity. Both the camera and object are more likely to move along this axis, unless the moving object is falling.

What will provide the better result when applying the filter? Filtering on one of the velocities, perhaps the one with consistently larger movement? Or perhaps both?

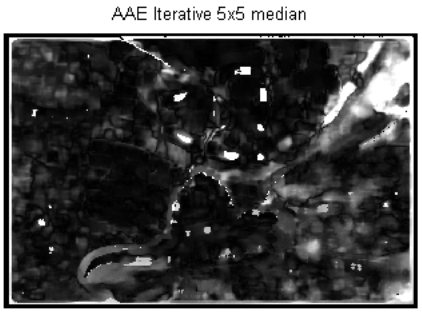
Image	AAE	AEE
Applying weights on V:		
Dimetron	10.0989	0.4899
Hydrangea	11.3818	1.0884
RubberWhale	14.5701	0.4469
Applying weights on U:		
Dimetron	11.6019	0.5252
Hydrangea	14.4055	1.2074
RubberWhale	17.0707	0.5120
Applying weights on U and V:		
Dimetron	10.3814	0.4800
Hydrangea	13.1699	1.1336
RubberWhale	15.5772	0.4570

Table 4.5: Table of the error change from weighting different velocities.

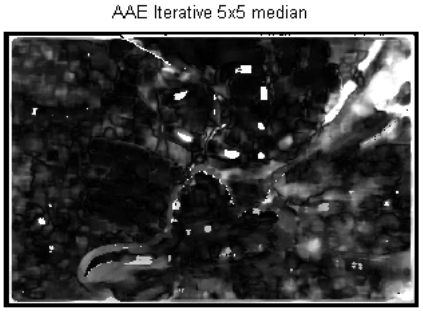
The table 4.5 display that just weighting the V velocity is a good idea. This means that change in Y-direction should be filtered for optimal effect.

Figure 4.4 display the angular error of the computed Dimetron sequence. The different images have been weighted accordingly, 4.4a on the U velocity. 4.4b on the V velocity and 4.4c on both U and V velocities. Weighting on both the velocities seem to help the surrounding area of the object. However the table 4.5 dispute this, and inform of an increased average error.

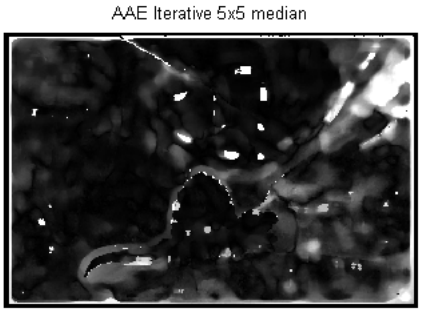
The X-direction usually have the most movement as few object move along the Y-axis, unless falling. The X-direction should have more movement, which is why it is harder to filtrate in this direction. The Y-direction should have less movement, and a sudden outlier is easily filtrated. This is probably why the median filter have more effect on this velocity.



(a) Weighted U.

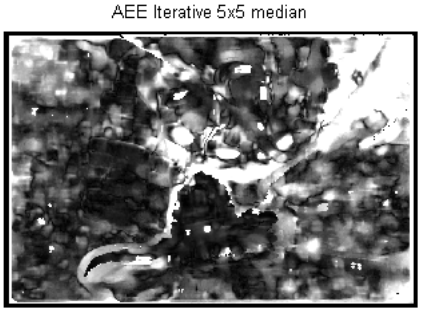


(b) Weighted V.

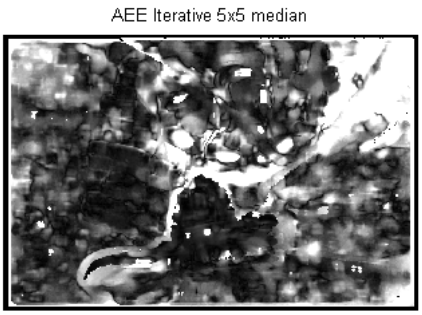


(c) Weighted U and V.

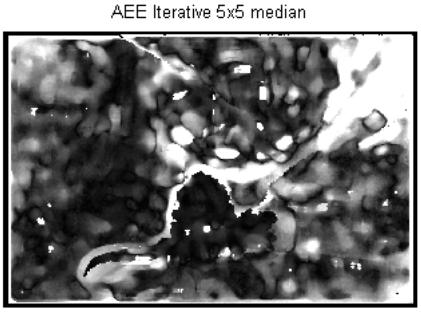
Figure 4.4: Displaying the Angular error of different weights on U, V and UV velocities.



(a) Weighted U.



(b) Weighted V.



(c) Weighted U and V.

Figure 4.5: Displaying the endpoint error of different weights on U, V and UV velocities.

The endpoint is precise with the dimetron, however it struggles with the surrounding

areas. The diffuse reflection point of the image is consistently wrong on all images.

Both the angular and endpoint in figure 4.4 and 4.5 struggles with the area surrounding the tail and the diffuse reflection of the cloth.

The figures 4.4 and 4.5 display small differences between weighting on V velocity or UV velocity. The table 4.5 is rendered credible, and the V velocity should be added median filtrated weights.

4.4 Hierarchical LK-method

The three sequence used 75 seconds each to compute the velocities. The run time of the iterative LK method was approximately 30 seconds. The first impression of the computation warn us that the iterative is more precise. It does not struggle with outliers and single point error to the same degree that the Hierarchical computation does. A weighted median should perform better on the hierarchical method because of this.

The first weight function tested was the sigmoid function from figure (4.2). This function deliver high weights to corners and some to flat regions.

AAE HLK	AAE HLK weighted	AEE HLK	AEE HLK weighted
Dimetron Image sequence:			
16.7625	14.5239	0.9203	0.7456
Hydrangea Image sequence:			
14.9308	12.1603	1.4641	1.1297
RubberWhale Image sequence:			
21.8339	17.1046	1.0141	0.7118

Table 4.6: The error of the weighted median filter applied on the Hierarchical LK-method

Brightness of pixels is determined by the response of the camera to light, the fraction of light reflected from the surface to the camera and the amount of light falling on the surface [13].

Table (4.6) and figure (4.10) both shows that the hierarchical computation is improved by a weighted median filter.

4.4.1 Dimetron Image sequence

Figure 4.6 marks the most obvious problem areas of the Dimetron sequence. Figure 4.7a illustrates how these area change from the hierarchy computation to the hierarchy computation with a weighted median filter (4.7b).

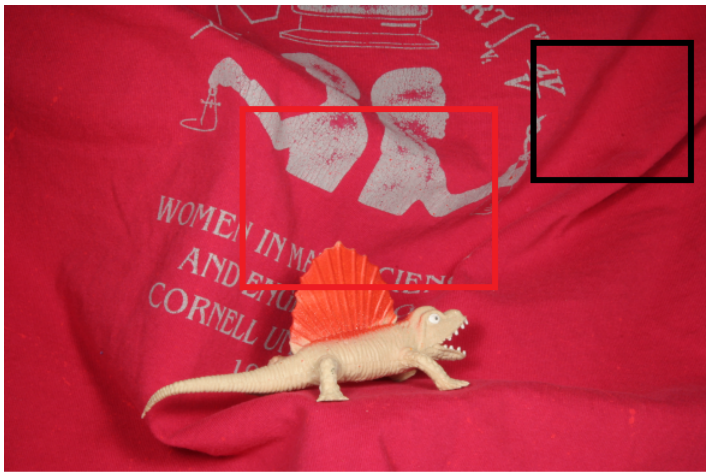
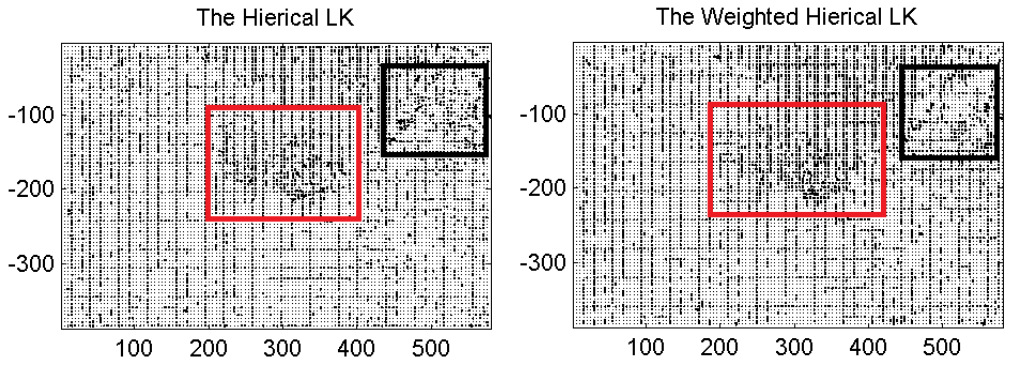


Figure 4.6: The Original image.



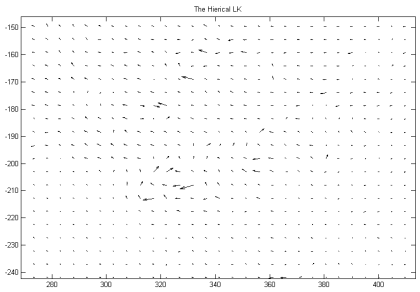
(a) Hierarchical Dimetron.

(b) weighted Hierarchical Dimetron

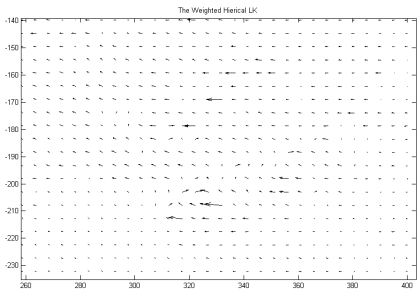
Figure 4.7: The troubled area of the Dimetron image sequence.

The Original image sequence from figure 4.6 show that these problem areas are in regions of diffuse reflection. With a changing image these intensities are expected to vary more, and error can easily occur here. Figure 4.8 illustrates the problem area of the Dimetron sequence from figure 4.7 with diffuse reflection. The surface reflects light, and scatters light evenly across the directions leaving a surface. The brightness does not depend on the viewing direction.

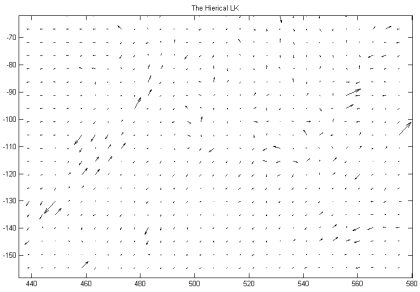
Figure 4.7 and 4.8 illustrated how the weighted median affect the flowfield. The filter is effective on the angle, however the error vectors still struggles with large endpoint values. From these plots, the improvement on the endpoint is still unclear. The histogram from figure 4.9 illustrates this better.



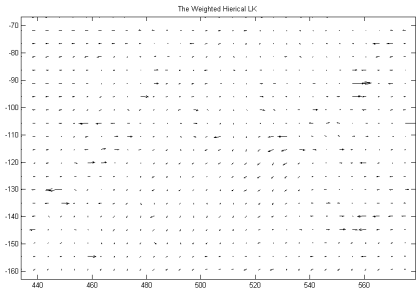
(a) The area marked with red from figure 4.7a.



(b) The weighted plotted marked with red in the original plot figure 4.7b.



(c) The area marked with black in figure 4.7a.



(d) The weighted plot of the area marked with black in figure 4.7b.

Figure 4.8: The plotted troubled area of the Dimetron image sequence.

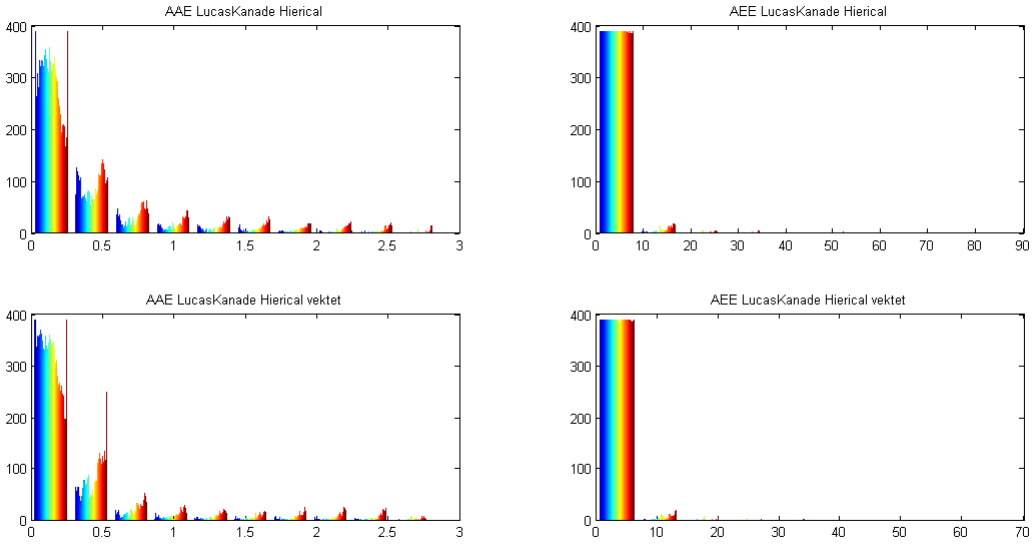
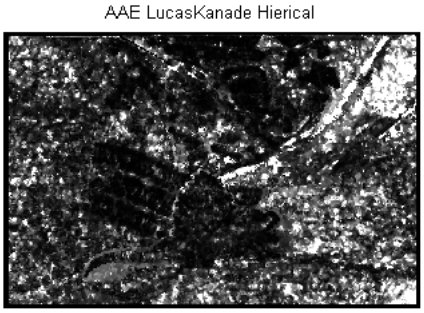


Figure 4.9: The histogram of the error in the Dimetron image sequence.

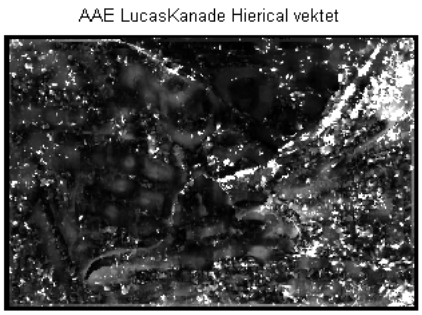
The x-axis on the AEE graph hints of improvements. A closer look on the data show that the biggest singular endpoint error is 87.4 high on the original hierarchical computation. The highest outlier in the weighted median is located at 41.1, half of the original computation.

In table 4.6 the Average Endpoint error goes from 0.92 to 0.74. While the Average Angular Error is changed from 16.7 degrees to 14.5 degrees. The density is 95.25%

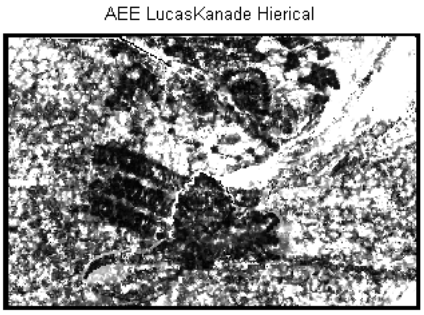
The body of the Dimetron, in the image sequence is clear, but it struggles with the area behind the Dimetron object head. The results from the error plot in figure 4.7 matches the black white error image in figure 4.10.



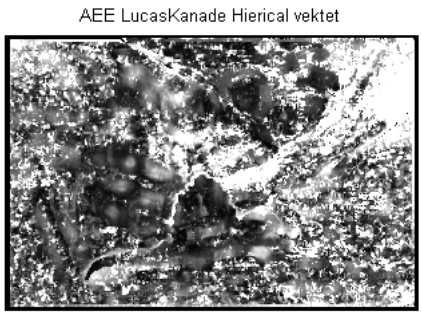
(a) AAE.



(b) AAE weighted.



(c) AEE.



(d) AEE weighted.

Figure 4.10: The Angular and Endpoint error of the Dimetron image sequence.

4.4.2 Hydrangea Image sequence

Hydrangea is part of the hidden texture sequence from the middlebury benchmark. The petals on the flower is dominated by shadows and detailed texture. If the surface cannot see the light source it is defined as a shadow. The surface is not dead black, because the surface is interreflected from other surfaces. These parts are directed away from the illumination direction.

The marked area is a very detailed area with some degree of shadow. Movement here are harder to track, which is why the problem area are the petals of the surrounding hydrangea. The pixels are harder to track because of the very detailed petals which are prone to change. Optical flow is known to struggle with among other things, shadows. This leads to outliers caused by lighting change and occlusions.

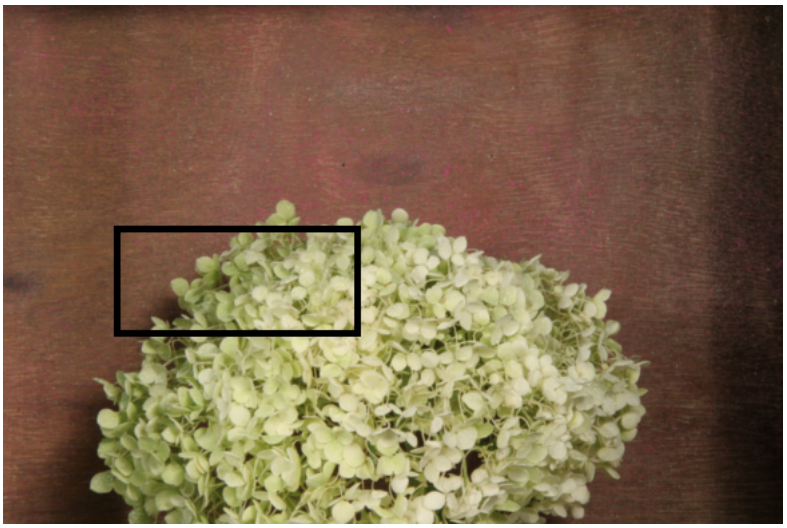


Figure 4.11: The Hydrangea image.

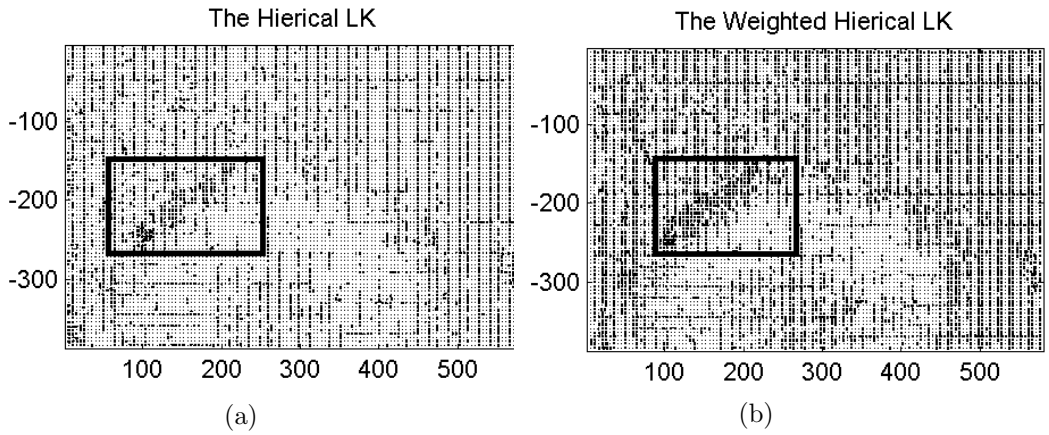
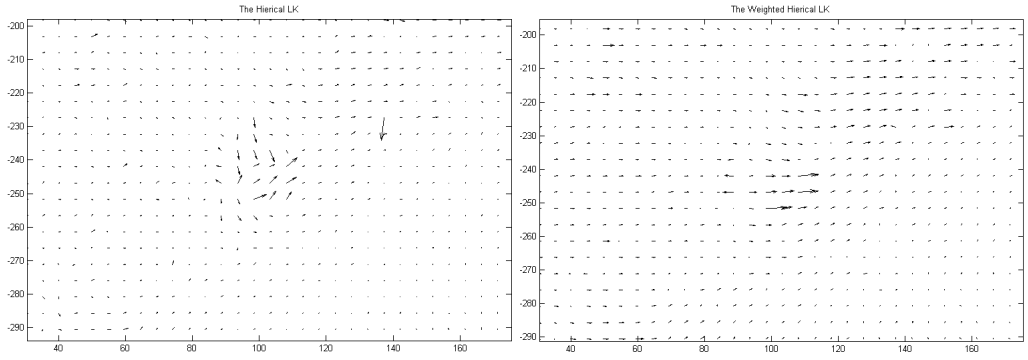


Figure 4.12: The Hydrangea image sequence

Figure 4.13 illustrates the same as the previous chapter showed on the Dimetron sequence. The median filter changes the angle of the flow, but still struggle with large endpoint error.



(a) (b)

Figure 4.13: Plot of point X:100 Y:240

The hierarchical computation struggles with outliers. The vector in X:140 Y:325 from figure 4.13 is an example of this. The weighted median filtrated removes this outlier and is replaced with the weighted median of the neighbourhood.

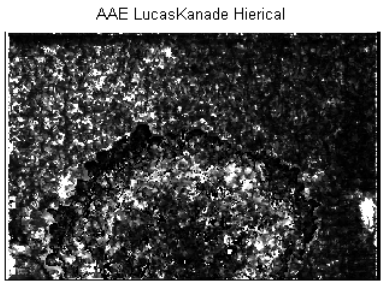
The error is still high in the plotted area. The weighted median filtrates on the angle, but the endpoint is to dominated by the neighbourhood and remains large.

This area in the left corner of figure 4.14d is still dominated by white regions, illustrating region error.

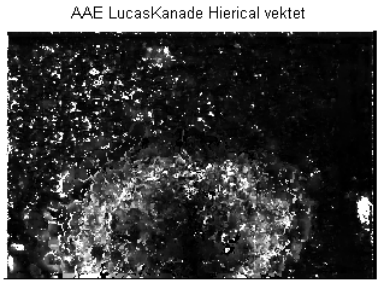
In this image sequence, the endpoint error surrounding the hydrangea is more dominated by error after applying a post processing step.

The histogram of the error in figure 4.15 represent the angular and endpoint error of the estimates. Figure 4.15 is harder to interpret, because of the large X-axis. A closeup informs of the largest single endpoint error is reduced from 121 to 110. The angular error had a collection of error on around 2.9 degrees. The weighted median filter actually increased this number of angular error from 30 to 42 vectors.

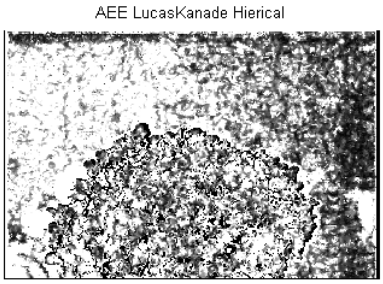
From table 4.6 the average error on the hydrangea is reduced from 14.9 to 12.1 on average angular error, and 1.4 to 1.2 on the endpoint error. This seems more lika an improvement than the figure 4.15 and 4.14 represents. The average might be improved but some regions still struggles with error.



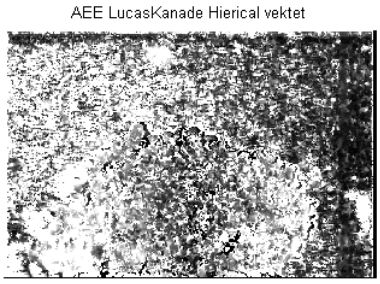
(a) AAE.



(b) AAE weighted.

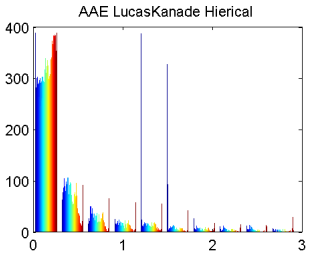


(c) AEE.

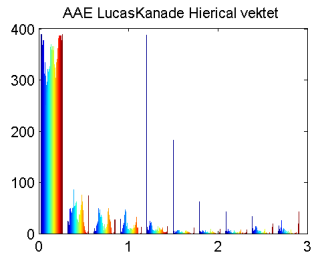


(d) AEE weighted.

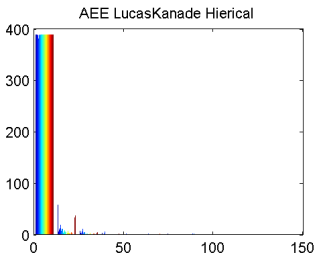
Figure 4.14: The estimated angular and endpoint error of the Hierarchical LK-method



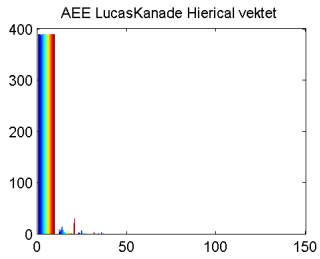
(a) Average angular error



(b) Average angular error with weights



(c) Average endpoint error



(d) Average endpoint error with weights

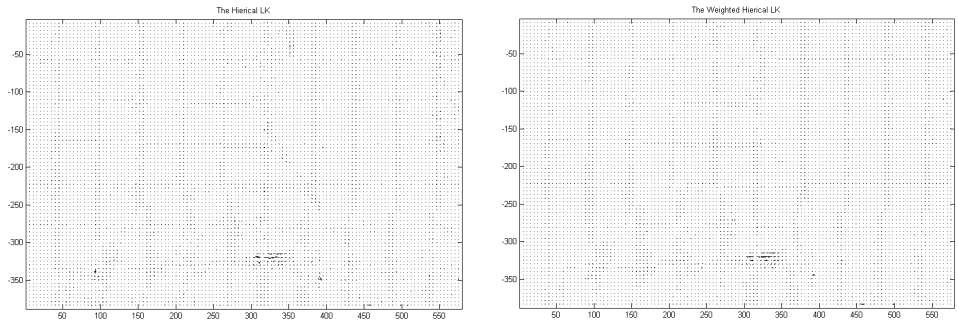
Figure 4.15: Histogram of error

4.4.3 RubberWhale

The RubberWhale sequence is dominated by small vectors. Movements are small and large flow in this image scene is either an error or an outlier.

Figure 4.16: The rubberwhale image sequence.

The details of the image sequences are many with individual moving objects, like letter objects and a rubberwhale toy. There is also texture in the form of different cloths and a fence like background. A sea shell also dominates the foreground of the image, and judging from the error in figure 4.20, this is more easily computed than the rest.



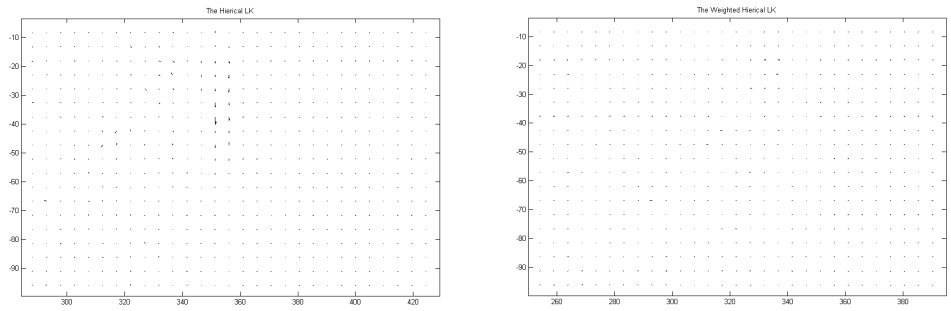
(a) Hierarchical LK (b) Weighted hierarchical LK

Figure 4.17: Plot of the RW vector field

The plotted vector field in figure 4.17 illustrates a region of large movement around the cardboard box.

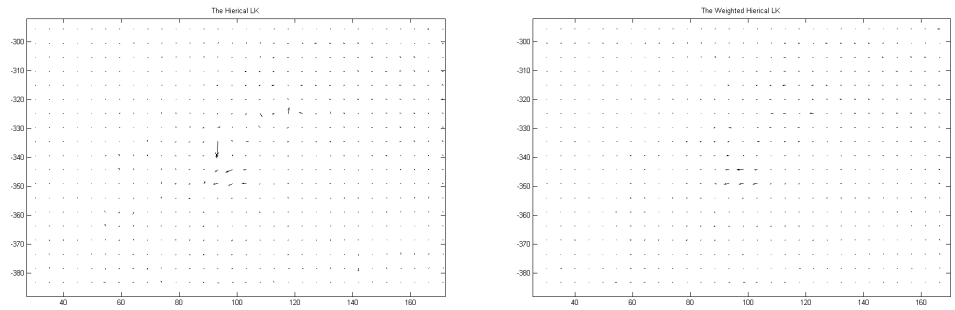
Figure 4.18 illustrates an outlier and a small region of error. The outlier is removed in the weighted computation, but the endpoint of the region remain large. This error centred around the cardboard, making it possible the error is created from texture confusion.

Figure 4.19 is a close-up of the fence in the background, creating confusion when the texture changed. The endpoint of the region remain large after the computation due to weights.



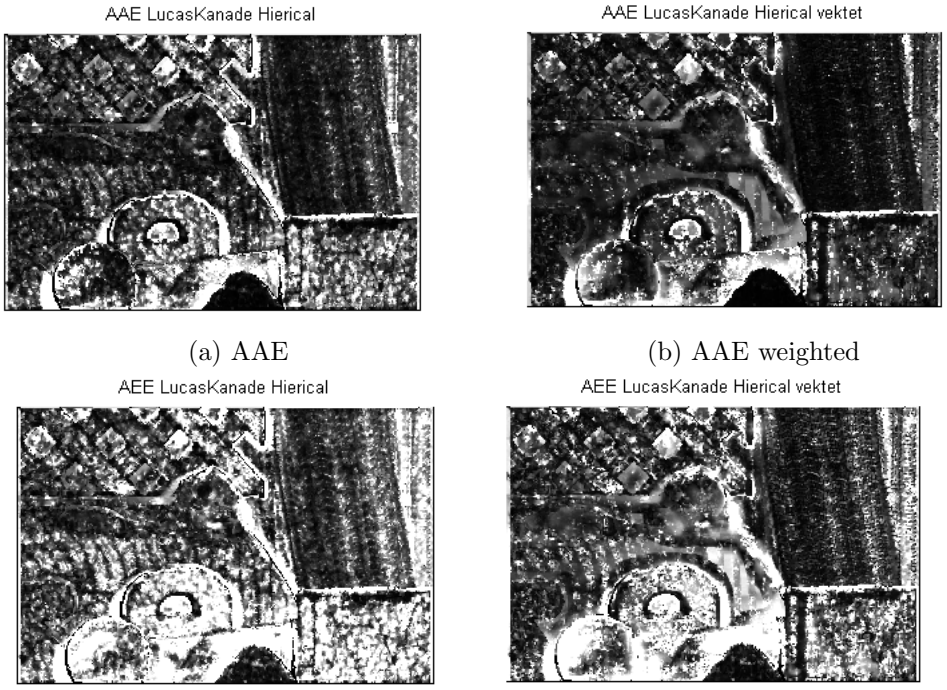
(a) Hierarchical LK (b) Weighted hierarchical LK

Figure 4.18: Outake of X:320 Y:50



(a) Hierarchical LK (b) Weighted hierarchical LK

Figure 4.19: Outake of X:100 Y:340



(a) AAE (b) AAE weighted

(c) AEE (d) AEE weighted

Figure 4.20: The Angular and Endpoint error of the RubberWhale image sequence

The most pronounced error is centred around the card box from figure 4.20. The letter objects D and O also struggles with small parts. Only the seashell is consistently accurate in both computation. The error struggles more with the blue striped cloth, than the detailed crotched fabric.

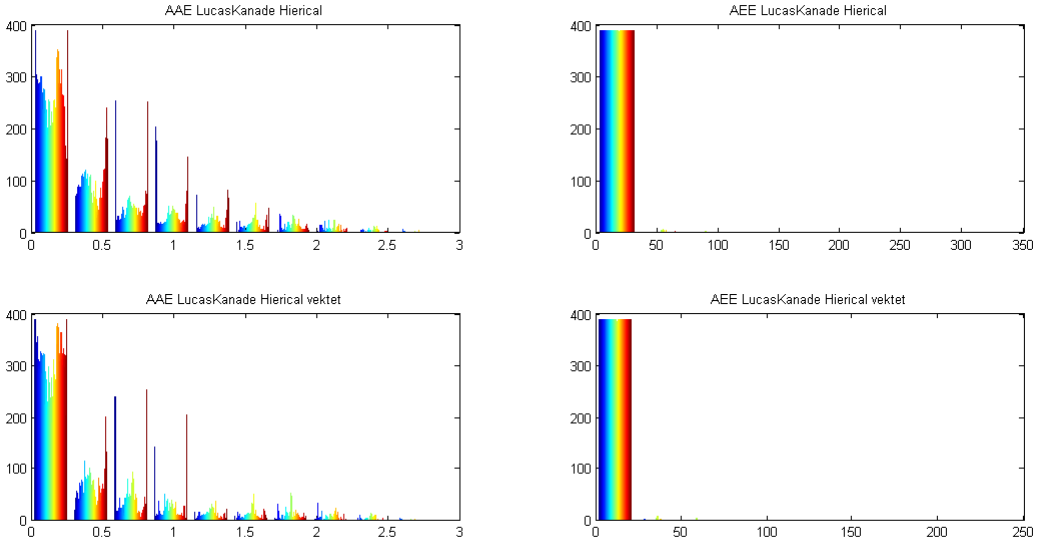


Figure 4.21: Histogram of angular and endpoint error.

The histogram in figure 4.21 illustrated the angular and endpoint error of the image sequence. The angular error is increased from the original computation to the weighted estimation. This is unfortunate.

4.5 Discussion

The success of Sun, Roth and Black was linked to the relationship between median filtering and the L1 penalty function of Li and Osher. The median filter de-noised during optimization and it lead to higher energy solutions. In their thesis they explain this to a deeper extent [35].

Instead of a penalty function, the LK-method uses a LS approach to minimization. This method have no relationship to the L1 penalty function, and the median filter showed no significant improvement of the iterative flow field of Lucas-Kanade. The iterative struggled more with regions of error, in the background of scenes. The median filter mask is overwhelmed and end up smoothing based on the regions.

The median filter was the key to remove outliers in the hierarchical LK-method. The weights were distributed from the scene, making sure that the median did not over filtrate on under represented vectors located at corners.

Surprisingly the LK-method struggled with flat regions and produced large angular and endpoint error in all image sequences.

The weighted median filter did improve accuracy and reduce error, however this filter is not the solution to optimization. If other optical flow estimation methods struggles with singular error, the median will work just as fine of the flow field, as it did on the LK-method.

5. Conclusion

The main challenge of the implementation was sensitivity. Adding a simple step, like smoothing and changing the derivative filter to compute the gradient had a great impact of the accuracy of the flow field. The goal was to remove outliers, vectors lost due to illumination change of partial occlusions.

The nature of the median filter is to sort the values in a mask and find the median of the neighbourhood. The center pixel of the mask is replaced with the new median of the mask. Weights were introduced to ensure that under represented pixels in corners was not over filtrated. To ensure this does not happen, several weight functions were tested. A step function chosen to distribute weights to corners.

The results showed that the median filter removed outliers in all cases. A single misplaced error in a the neighbourhood was replaced with the median of the mask.

The median filter struggled with regions of high error, caused by among others, illumination changes in the image. In most cases the angle of the vector was restored, but the endpoint of the region vectors was still high.

The image scenes had challenges with illumination change such as diffuse reflection and shadows, both sensitive to lighting change. The scenes also had transparent materials, like the crotched cloth of the RubberWhale sequence, causing error. These materials was often the cause of error.

The iterative LK-method struggled less with single point outliers and more with regions of error. The median did little to improve this, because the median will work best on single point error.

This was also the case for the hierarchical method, where the weights were applied to a window mask, moving though the estimated flow. The weights were chosen to filtrate on only the V vector, representing motion in Y-direction. The weights seemed to overfiltrate when applied to both velocities. Further works would include trying to filtrate both velocities, or filtrate on the angle and/or endpoint of the vector. The filter did improve

on the angle. Filtering on the endpoint errror could reduce the error of regions.

The computational complexity was not a problem since a real time estimation was not a goal. The iterative used approximately 20 seconds against the hierarchical time of 75 seconds. The gain of accuracy against computational complexity was not solved in this thesis.

The median filter worked well on singular error such as an outlier, but struggled with regions of error, making the error the median of the filter mask.

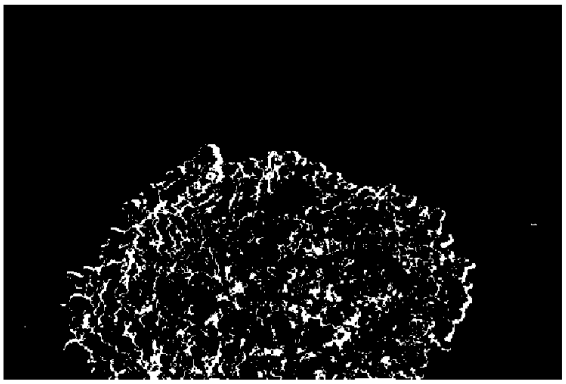
The weighted median is a simple solutions for optical flow method struggling with singular error. The filter will work well for other methods, not just Horn-Schunck and Lucas-Kanade.

List of Figures

List of Tables

A. Density Images

(a) The Dimetron Density



(b) The Hydrangea Density

(c) The RubberWhale density

Figure A.1: Density; Dimetron, Hydrangea and RubberWhale

B. Zip File

The files in the zip-file include the function:

- colormap
- colorTest
- computeColor
- Expand
- flow_ aae
- flowAngErr
- flowAngErrUV
- flowToColor
- HiarchicalLK
- LucasKanade
- LucasKanadeRefined
- LucasKanadeVektet
- main
- mainVektet
- plotError
- plotflow
- readFlowFile
- Reduce

- vektet_ median
- writeFlowFile

The images:

- Dimetrodonflow10
- Dimetronframe11
- Hydrangeaframe10
- Hydrangeaframe11
- RubberWhaleframe10
- RubberWhaleframe11
-
-

The flo-files for ground truth:

- Dimetrodonflow10
- Hydrangeaflow10
- RubberWhaleflow10

Bibliography

[1] "vision.middlebury.edu/flow/".

[2] I. Austvoll. *Motion Estimation using Directional Filters.* PhD thesis, PhD thesis, NTNU/HIS, PoBox 2557 Ullandhaug, N-4091 Stavanger, Norway, 1999.

[3] Andrew Bainbridge-Smith and Richard G Lane. Determining optical flow using a differential method. *Image and Vision Computing*, 15(1):11–22, 1997.

[4] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[5] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[6] JL Barron and NA Thacker. Tutorial: Computing 2d and 3d optical flow. *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, 2005.

[7] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[8] T.Kanade B.D.Lucas. An iterative image registration technique with an application to stereo vision. *Proceedings of imaging Understanding Workshop*, 1981.

[9] Michael J Black and Paul Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer vision and image understanding*, 63(1):75–104, 1996.

[10] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[11] William J Christmas. Filtering requirements for gradient-based optical flow measurement. *IEEE Transactions on Image Processing*, 9(10):1817–1820, 2000.

[12] D.Fleet. *Measurement of Image Velocity*. Kluwer Academic Publishers, 1992.

[13] J.Ponce D.Forsyth. *Computer Vision, a modern approach*, chapter 5, pages 171–178. Pearson Education Limited, second edition, 2012.

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] David Fleet and Yair Weiss. Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*, pages 237–257. Springer, 2006.

[16] Darya Frolova and Denis Simakov. *Matching with Invariant Features*. The Weizmann Institute of Science, March 2004.

[17] Takashi Fuse, Eihan Shimizu, and Morito Tsutsumi. A comparative study on gradient-based approaches for optical flow estimation. *International Archives of Photogrammetry and Remote Sensing*, 33(B5/1; PART 5):269–276, 2000.

[18] N Gautier and J-L Aider. Control of the separated flow downstream of a backward-facing step using visual feedback. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 469, page 20130404. The Royal Society, 2013.

[19] James J Gibson. The perception of the visual world. 1950.

[20] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[21] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[22] Berthold KP Horn. Understanding image intensities. *Artificial intelligence*, 8(2):201–231, 1977.

[23] I.Sobel and G.Feldman. *A 3x3 isotropic gradient operator for image processing*, volume Pattern Clas- sification and Scene Analysis, page 271–272. 1973.

[24] Joseph K Kearney, William B Thompson, and Daniel L Boley. Optical flow estimation: An error analysis of gradient-based methods with local optimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2):229–244, 1987.

[25] Sohaib Khan. University of Central Florida, http://crcv.ucf.edu/REU/reu2013/, May 2013. downloaded 16.04.2015.

[26] Yingying Li, Stanley Osher, et al. A new median formula with applications to pde based denoising. *Commun. Math. Sci*, 7(3):741–753, 2009.

[27] Vishal Markandey and Bruce E Flinchbaugh. Multispectral constraints for optical flow computation. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 38–41. IEEE, 1990.

[28] Hans-Hellmut Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21(1):85–117, 1983.

[29] Atsushi Nomura, Hidetoshi Miike, and Kazutoshi Koga. Field theory approach for determining optical flow. *Pattern Recognition Letters*, 12(3):183–190, 1991.

[30] Xiaofeng Ren. Local grouping for optical flow. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[31] R.Fischer, S.Perkins, and E.Wolfart. Pixel values. http://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm, 2003.

[32] Nusrat Sharmin and Remus Brad. Optimal filter estimation for lucas-kanade optical flow. *Sensors*, 12(9):12694–12709, 2012.

[33] Eero P Simoncelli. Design of multi-dimensional derivative filters. In *ICIP (1)*, pages 790–794, 1994.

[34] Lee Yee Siong, Siti Salasiah Mokri, Aini Hussain, Norazlin Ibrahim, and Mohd Marzuki Mustafa. Motion detection using lucas kanade algorithm and application enhancement. In *Electrical Engineering and Informatics, 2009. ICEEI'09. International Conference on*, volume 2, pages 537–542. IEEE, 2009.

[35] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[36] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014.

[37] Deqing Sun, Stefan Roth, JP Lewis, and Michael J Black. Learning optical flow. In *Computer Vision–ECCV 2008*, pages 83–97. Springer, 2008.

[38] O Tretiak and L Pastor. Velocity estimation from image sequences with second order differential operators. In *Proc. IEEE ICPR, Montreal*, pages 20–22, 1984.

[39] Sergio Uras, Federico Girosi, Alessandro Verri, and Vincent Torre. A computational approach to motion perception. *Biological Cybernetics*, 60(2):79–87, 1988.

[40] Robert J Woodham. Multiple light source optical flow. In *Computer Vision, 1990. Proceedings, Third International Conference on*, pages 42–46. IEEE, 1990.

[41] Minghao Yang, Jianhua Tao, Lihui Shi, Kaihui Mu, and Jianfeng Che. An outlier rejection scheme for optical flow tracking. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–4. IEEE, 2011.