MITIGATION OF ATMOSPHERIC TURBULENCE DISTORTIONS IN LONG RANGE VIDEO SURVEILLANCE

P.E. Robinson* and W.A. Clarke**

* *Faculty of Engineering and the Built Environment, University of Johannesburg, PO Box 524, Auckland Park 2006, South Africa*
E-mail:philipr@uj.ac.za
** *SAIEE, IEEE, Faculty of Engineering and the Built Environment, University of Johannesburg, PO Box 524, Auckland Park 2006, South Africa*
E-mail:willemc@uj.ac.za

Abstract: This paper explores the problem of atmospheric turbulence in long range video surveillance. This turbulence causes a phenomenon called heat scintillation or heat shimmer which introduces distortions into the video being captured. The nature of these distortions is discussed and a number of possible solutions explored. Using these solutions, three algorithms are implemented to attempt to mitigate the effects of heat shimmer. Within this field there is very little subject matter on the topic of objectively comparing the performance of heat shimmer reduction algorithms. A set of possible metrics is proposed in this paper and used to compare the performance of the implemented algorithms. These results provide insight into the nature of the algorithms and the effectiveness of the metrics under consideration.

Keywords: Atmospheric Turbulence, Scintillation, Heat Shimmer, Terrestrial, Optical Flow, Deblurring, Quality Metrics, GPU.

1. INTRODUCTION

The rapid advancement of digital image capture technology has resulted in the development of many technologies devoted to making use of these captured image and video signals [1, 2]. These signals can be captured in a variety of circumstances and more often than not contain a variety of distortions and noise. This degradation is the result of various factors, including the nature of the hardware used to capture the signal and external environmental factors such as lighting and camera jitter [2]. These signals are often used not only by a human operator who could deal with poor quality video, but also automated processing systems that make use of the video signal to automatically gather information about the scene. For these systems quality degradation is a serious problem [1, 2, 3].

Imaging systems are capable of very high magnifications. These kinds of systems are used for long range video surveillance. In the case where the target scene is being viewed from a range of more than 1 km the effects of atmospheric turbulence become apparent [1, 4].

Turbulence in the atmosphere causes pockets of air of varying temperatures and thus densities to move in a random fashion. This movement is caused by the varying densities of the air pockets, wind and terrain. Light from the target scene must travel through this turbulent atmosphere to reach the imaging system. The varying densities of the air pockets cause this light to be refracted by varying degrees and in a continually changing manner. This results in the target scene appearing blurred and to be wavering or shimmering. This means that objects in the scene will appear to be moving even when stationary. This effect is dubbed heat shimmer [1, 3, 4, 5, 6, 7].

Heat shimmer severely limits the range of long range imaging systems, and as such, mitigating the effects of atmospheric turbulence is a major concern when designing these systems [1, 4, 6, 7].

There are two main schools of solution to this problem. The first is to make use of a mechanical adaptive optics system to physically compensate for the effects of the atmospheric turbulence on incoming light rays [11]. The second solution is to make certain assumptions about the nature of the distortion and make use of an image processing approach to digitally enhance the video signal to attempt to reduce the distortion. A few proposed image processing methods are the direct Discrete Fourier Transform (DFT) solution [4], image registration and fusion [1], Adaptive Control Grid Interpolation [6, 7], Image Time Sequence Registration [8], Neural Network approach based on the Monte Carlo [9] and Homomorphic and Power Spectrum approach [10].

While a number of techniques have been proposed to deal with the problem of heat shimmer there has been little work done to find methods of comparing the effectiveness of these techniques. This is primarily due to the fact that it is very difficult to obtain ground truths of real world heat shimmer distorted video. This paper describes the implementation of three algorithms that mitigate the effects of heat shimmer. A number of performance metrics are proposed for use in comparing the effectiveness of these algorithms. Using the proposed metrics the performance of the implemented algorithms are compared. The results will show which of the proposed metrics are viable measures of performance for these types of algorithms.

The other focus of this paper is to start on the road to developing a real-time implementation of a heat shimmer mitigation algorithm. These algorithms are very computationally intensive and there is no account in the literature of how a real-time implementation could be achieved. This paper will explore the use of Graphics Processing Units (GPUs) as tools to accelerate the processing time of these types of algorithms.

The remainder of this paper will be arranged as follows. Section II will give an overview of the implemented algorithms. Section III will describe the details of a GPU implementation of one algorithm. Section IV will describe the proposed performance metrics and Sections V and VI will described the experiments and results. Finally VII will present the conclusion.

2. Overview of Algorithms

The algorithms implemented in this paper classify the effects of heat shimmer on long range video into two forms of distortion as described in [6, 7]. The first effect of heat shimmer is to cause photometric distortion or blurring of the scene being viewed. The second effect is geometric distortion which produces an apparent wavering motion of elements in the scene. An assumption is made that the geometric distortion is quasi-periodic. This means that any given point in the scene should oscillate periodically around its true position. The algorithms in this paper seek to reduce the effects of heat shimmer by tackling each of the above described distortions.

1.1 Averaging and Wiener filtering algorithm

This algorithm was developed in conjunction with [23] with the intention of developing the simplest possible technique to mitigating the problem of heat shimmer. The algorithm tackles each of the distortions introduced into the captured video by heat shimmer independently. Firstly the algorithm seeks to reduce the amount of geometric distortion in the video sequence by exploiting the assumption that the geometric distortion is quasi-periodic. To do this a simple ratio averaging scheme is used to maintain a running average of the frames in the sequence as shown in the following equation:

$$g(x,y,t) = \alpha f(x,y) + (1-\alpha) g(x,y,t-1), \tag{1}$$

Where:
x, y = the pixel coordinates in an image frame
t = the current frame in the sequence
g(x,y,t) = updated ratio frame average
g(x,y,t-1) = the previous frame average
f(x,y) = the current incoming frame
α = a scalar value between 0 and 1

α is chosen for the algorithm and controls the weighting of the ratio of the current frame and the previous frame average that appears in the new frame average. The quasi-periodic motion in the sequence is thus averaged out to zero. The down side of this process is that the averaging operation introduces blur into the sequence in addition to the photometric distortion already present.

The second stage of the algorithm addresses the photometric distortion. This step makes use of a model of the blurring effect of the atmosphere and a Wiener filter that allows one to filter out the blur represented by this model from the video sequence to produce a sharper image. The following equation shows the transfer function of the Wiener filter [2, 13]:

$$H_w = \frac{H^*}{|H|^2 + {}^{S_N}/_{S_F}}, \tag{2}$$

Where:
S_N = power spectrum of the additive noise
S_F = power spectrum of undistorted the image
H = the model of the distortion
H_w = the Wiener Filter based on the specified model

The modulation transfer function (MTF) used as a model for this process was presented by Hufnagel and is shown below [12]:

$$H(u,v) = e^{-\lambda(u^2+v^2)^{5/6}}, \tag{3}$$

Where:
λ = parameters that controls the amount of blur present in the model
u and v = the 2 dimensions of the spatial frequencies.

The noise-to-blurred signal ratio is estimated as the difference between the global variance of the whole frame and the average local variance. This is due to the fact that the noise and the blurred image are uncorrelated. The NSR is thus estimated using the following equation:

$$NSR = \frac{Local\ Variance}{Global\ Variance - Local\ Variance}, \tag{4}$$

The primary issue with this process is that the nature of blurring effect of the atmosphere is completely unknown and needs to be determined using only the distorted video sequence.

This is done by defining a search space of plausible values for λ and applying the Wiener filter to the current frame for each of these values and then using a metric to determine which of the λ values results in the sharpest output. The sharpness metric used to measure the sharpness of the resulting frame is based on the Laplacian operator which is defined as follows [14]:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}, \tag{5}$$

Where:
I = the 2-dimensional intensity map of the image.

The effect of this operator can be approximated by the following discrete kernel [11].

-1	-1	-1
-1	+8	-1
-1	-1	-1

Figure 1: Laplacian operator approximation kernel

This kernel is applied to the image and accentuates areas of high spatial frequencies such as edges and reduces the intensities of areas of uniform colour. After the operator is applied the average intensity is found of the result. The more high frequency content there is in the image the higher this metric will be and thus the sharper the image appears. This allows us to find which value of λ resulted in the sharpest image and use that frame as the output of this stage of the algorithm. In [7] it is proposed to use the kurtosis of the image as a sharpness metric.

Both the sharpness metric mentioned above are time domain metrics which is a disadvantage when working with the Wiener Filter as the Wiener Filter is a frequency domain filter. This means that an incoming frame needs to have the Fast Fourier Transform (FFT) performed on it before it can be filtered. Then after the frame is filtered for each value of λ it will need to have the Inverse Fast Fourier Transform (IFFT) performed before the time domain sharpness metrics can be used.

It is proposed that a frequency domain sharpness metric be used so that only one FFT and IFFT operation needs to be performed per frame. The frequency domain sharpness metric designed for this task measures the power present in the high frequency bands of the image. This is done by finding the magnitude of the FFT values in the area described in figure 2.

In the figure the DC value is assumed to be at the centre. The range of frequency values measured fall into the light grey area described by the normalized coordinates. This area of measurement was found experimentally to give the most reliable sharpness measurements. The flow diagram of the entire algorithm is presented in figure 3.

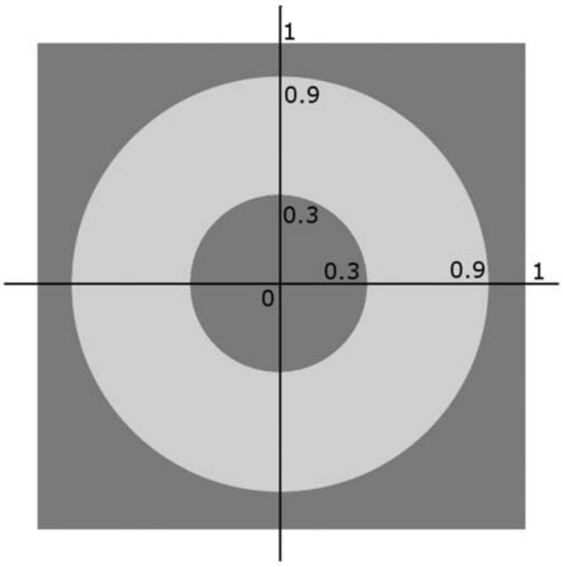

Figure 2: Frequency domain sharpness metric area of measurement. DC value assumed to be at centre.

2 WIENER FILTERING AND OPTICAL FLOW BASED DEWARPING ALGORITHMS

The following two algorithms are based on the work of [6, 7] and are chosen due to the fact that they follow the same paradigm as the first algorithm. This paradigm treats the effects of heat shimmer as two separate distortions, photometric and geometric. These approaches first deal with the photometric distortion and adhering to the same paradigm means we can make use of the Weiner Filtering and Sharpness metric based method used in the first algorithm.

These algorithms however use a different approach to dealing with the geometric distortion as suggested in [6, 7]. An optical flow technique is used to map the movement of all the pixels from one frame to the next in the video sequence. Optical flow techniques produce a vector field that describes the motion of all the pixels between two video frames. The two optical flow algorithms implemented were the Lucas-Kanade [14] and Horn-Schunk [15] algorithms, which are the most common optical flow techniques. Both of these algorithms make use of differential methods to determine the optical flow between two frames.

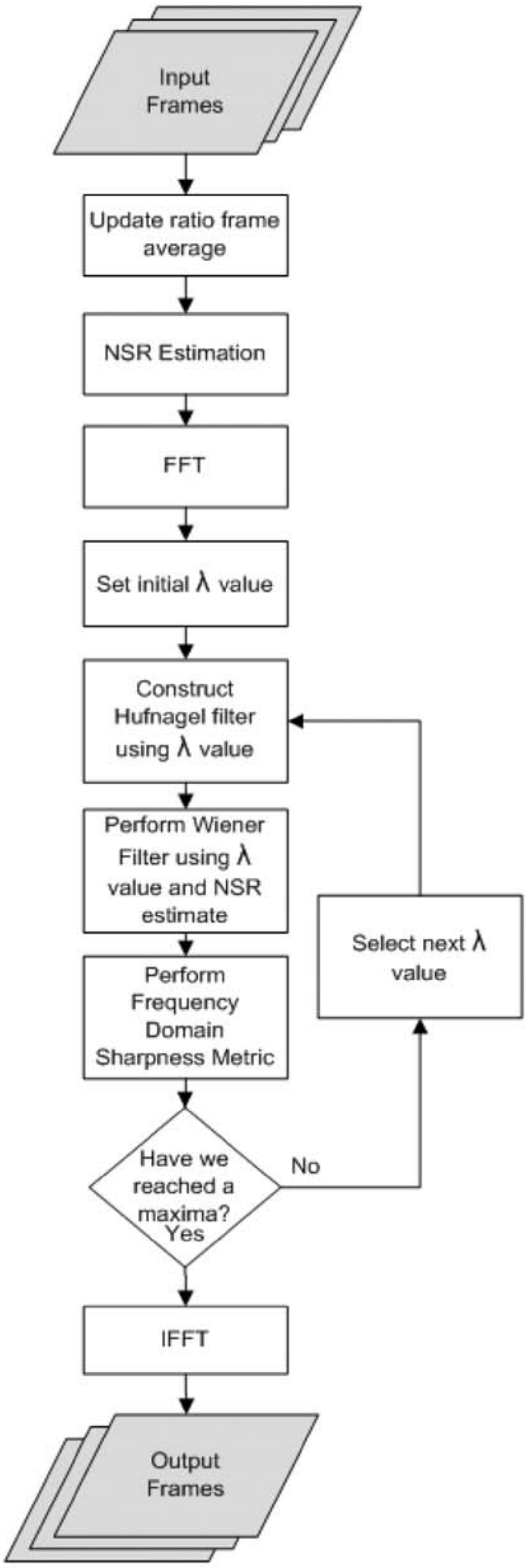
Input Frames
Update ratio frame average
NSR Estimation
FFT
Set initial λ value
Construct Hufnagel filter using λ value
Perform Wiener Filter using λ value and NSR estimate
Perform Frequency Domain Sharpness Metric
Have we reached a maxima?
No
Yes
Select next λ value
IFFT
Output Frames

Figure 3: Averaging and Wiener filtering algorithm block diagram

The general assumption made by these algorithms is that while pixels are moving from frame to frame, their intensities should not change significantly. Based on this assumption each algorithm defines a constraint equation that can be solved to find the optical flow for the given two frames which are captured at times *t* and *t+δt.* The motion for every pixel position is calculated by these techniques.

The Lucas-Kanade algorithm [14] makes use of the following constraint equation:

$$I(x,y,t) = I(x + \delta x, y + \delta y, t + \delta t), \tag{6}$$

Where:
$I(x,y,t)$ = the pixel intensity at the spatial coordinates x,y and the temporal coordinate t.
$\delta x,\ \delta y,\ \delta t$ = the distance that the pixel has moved between the two frames.

The equation states that the pixel intensity in the first frame will be the same at its new position in the second frame [14, 16]. Optical flow algorithms generally suffer from the Aperture problem, which means that the information contained within the two frames under consideration is not enough to solve for the velocity of every pixel in both the *x* and *y* dimensions. In the Lucas-Kanade algorithm this problem is overcome by assuming that the optical flow inside a small sub window of the frames centred around the current pixel under consideration is constant. By using a Taylor series expansion of equation 6 and using the least squares method, the following equation can be derived which describes how to calculate the *x* and *y* velocities of the pixel at the centre of the current window.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{y_i} I_{x_i} & \sum I_{y_i}^2 \end{bmatrix} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix}, \tag{7}$$

Where:
i = 1...n where n is the number of pixel elements in each window centred at x and y.
V_x, V_y = the velocities of a given pixel in the x and y directions.

We also define the following to represent the differentials used in this method:

$$\frac{\partial I}{\partial x} = I_x, \frac{\partial I}{\partial y} = I_y \text{ and } \frac{\partial I}{\partial t} = I_t, \tag{8}$$

The Horn-Schunk algorithm [15] is similar to the Lucas-Kanade method in that it makes use of the differentials between the two frames but it overcomes the Aperture problem by introducing a global smoothness constraint α.

The following equation is the energy function, in two spatial dimensions, that is minimized to find the optical flow [15]:

$$E = \int\int \left(I_x V_x + I_y V_y + I_t\right)^2 dxdy + \alpha \int\int \left\{ \left(\frac{\partial V_x}{\partial x}\right)^2 + \left(\frac{\partial V_x}{\partial y}\right)^2 + \left(\frac{\partial V_y}{\partial x}\right)^2 + \left(\frac{\partial V_y}{\partial y}\right)^2 \right\} dxdy \quad (9)$$

Using the Jacobi method to minimize this equation iteratively the following equations are found:

$$V_x^{k+1} = \overline{V_x^k} - \frac{I_x\left(I_x \overline{V_x^k} + I_y \overline{V_y^k} + I_t\right)}{\alpha^2 + I_x^2 + I_y^2}, \quad (10)$$

$$V_y^{k+1} = \overline{V_y^k} - \frac{I_y\left(I_x \overline{V_x^k} + I_y \overline{V_y^k} + I_t\right)}{\alpha^2 + I_x^2 + I_y^2}, \quad (11)$$

Where:
$\overline{V_x^k}$, $\overline{V_y^k}$ = the average of these velocity values in the local neighbourhood.

The Horn-Schunck method produces dense optical flow fields but is fairly sensitive to noise. The above techniques can be used to produce vector fields describing the motion of every pixel between two frames. If we make the assumption that the geometric distortion caused by heat shimmer is quasi-periodic [6, 7], we can use the optical flow data to find the true position of the pixels in a video frame. To do this for a specific frame, a window of frames around that frame is chosen and the optical flows between the current frame and each other frame in the window is calculated.

If the motion present in the frames is quasi-periodic, taking an average of the optical flow vectors should reveal the true position of each pixel as that pixel would have been wavering around its true position. This data is then used to move the pixels to their correct positions. This process can leave holes in the resulting image, which are filled in using the mean intensity in the current windows of frames at that position. The following figure shows the flow diagram of the algorithm.

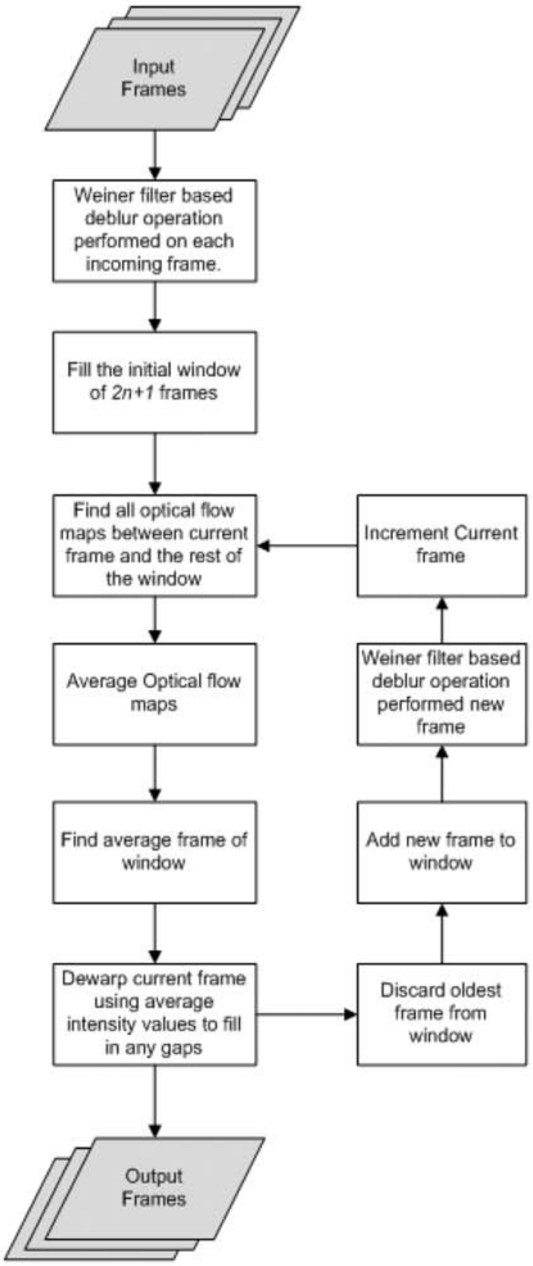
Input Frames
Weiner filter based deblur operation performed on each incoming frame.
Fill the initial window of 2n+1 frames
Find all optical flow maps between current frame and the rest of the window
Increment Current frame
Average Optical flow maps
Weiner filter based deblur operation performed new frame
Find average frame of window
Add new frame to window
Dewarp current frame using average intensity values to fill in any gaps
Discard oldest frame from window
Output Frames

Figure 4: Wiener filtering and Optical flow based algorithm block diagram

3 GPU IMPLEMENTATION

Image processing technology is maturing very quickly and we have been seeing its real world application for a few years now. However, the standard CPU based computing platforms are still struggling to provide enough performance in order to implement image processing theory in a practical way. This is even true of the newest multi-core processors that are now available. This lack of performance has driven the use of non-standard platforms, such as the FPGA, for these tasks, and these platforms are expensive and difficult to develop for.

In recent years the Graphics Processing Unit (GPU) has emerged as a possible solution to this dilemma. GPUs are found on the commercial off-the-shelf graphics cards that we have been using for years for the express purpose of rendering graphics for video games. Recently GPUs have become user programmable and able to support floating point precision calculations. It has become apparent that for applications that can make use of their parallel stream processing architecture that GPUs can far outstrip CPU in price and performance.

A GPU can be seen as a parallel stream processor. This means that the GPU contains a number of processing pipelines each capable of processing an element of data independently and in parallel with the other pipelines. This architecture is useful when performing the same kernel or operation on each element of data in a set of data. As can be seen in Figure 5 this architecture has far less cache and control overhead than that of a traditional CPU. This is because each pipeline is self contained and once data is in the pipeline it does not need any further input until after the data emerges from that pipeline.

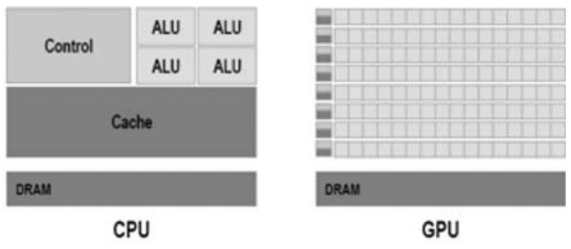

Figure 5: CPU vs. GPU architecture [17]

In the GPU data is represented as vertices, which are points in the co-ordinate system defined in the GPU, and textures or colours which are mapped onto the vertices. The graphics pipeline in a typical GPU is roughly summarised in figures 6 and 7 [18].

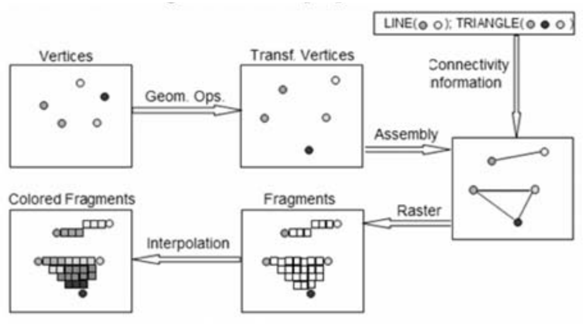

Figure 6: Visualisation of pipeline functions [18]

The parts of the pipeline that we are most interested in are parts whose functionality we can replace with our own programs. These are the vertex transformation stage and fragment texturing and colouring stage [18].

The process begins by taking the vertices, which are points in space that have colours, normals and texture coordinates associated with them, and transforming their characteristics. The next step is connecting each of the related vertices into lines and polygons. The next step is the step where the majority of the processing will be done. In this step the frame being rendered is rasterized. This means that the frame is divided into elements or fragments each representing a pixel in the frame. The texture and colour information of the vertices is then used to define the colour for each fragment in the scene. We will replace the fixed functionality with our own program. These programs are called the vertex and fragment shaders. The respective shaders are executed once for each vertex in the frame and once for each fragment in the frame [18].

The vertex and fragment shaders are invoked by rendering a quad, which is a rectangle with a vertex at each corner, which is exactly the size of the texture we want processed. The vertex shader is called once for each corner of the quad and the texture is rasterized in such a way that each fragment corresponds exactly to a pixel in the texture. Our fragment shader is executed exactly once for each of these fragments [18, 20].

For this project OpenGL was selected as the API used to interface with the GPU hardware. GLSL was selected as the shader scripting language. These selections were made primarily due to the platform portability of OpenGL and GLSL [19].

By default OpenGL will render its output to the screen but it is possible to use the FrameBuffer extension to redirect the rendered output to a frame buffer which has been linked to a texture. This allows one to use that output as the input for another rendering pass or to send that data back to the CPU after it has been processed [21]. This is critical for recursive algorithms.

The Lucas-Kanade based Wiener filtering and optical flow algorithm was implemented on the GPU to give an idea of the performance increases the GPU provides. The majority of the algorithm was easily translated into the parallel architecture of OpenGL as kernel convolutions are easily implemented to be executed for each fragment. The main issues encountered while porting the algorithms to OpenGL were the implementation of recursive algorithms, calculating the mean value of a texture and the dewarping stage of the algorithm.

The main recursive stage in the algorithm was the implementation of the decimation-in-time Fast Fourier Transform (FFT) [22]. The FFT is a recursive algorithm with a number of stages which are dependent on the previous iteration of the algorithm. This recursion is achieved using the ping pong techniques described in [21]. This uses two frame buffers which switch roles as the input to the current iteration and the storage space for the output of that iteration. This allows us to perform recursion with only two frame buffers.

While calculation of the mean of an image is relatively trivial in a serial architecture on a parallel architecture it presents a problem. A given shader program will be executed for each fragment of a texture independently and the shader program can not know the results of its execution on any other fragments during a same rendering step. This means there is no way to read the values of all the fragments in a way that will exploit the parallel nature of the GPU. To achieve the mean calculation in parallel a recursive down-sampling algorithm was used. In each successive stage of the algorithm the texture was down-sampled by a factor of 2 resulting in a texture a quarter the size of the original. This is repeated until the image is down-sampled into a single fragment which will be average value of the entire texture.

The most significant issue encountered during the GPU implementation was the lack of support for scattering operations in OpenGL 2. The Dewarping stage of the algorithm required that a given pixel be moved to a new location based on the average optical flow map. This is not possible using only fragment shaders as when a fragment shader is executed it can only affect the value of the pixel it was executed for. This means that it is impossible for a fragment shader to read from the average optical flow map where the current fragment should be moved to, and proceed to change the value of the destination fragment. This problem was solved by making use of the vertex shader. A grid of point vertices was created, one for each pixel in the texture to be dewarped. A Vertex shader was then written that is executed once for each vertex being rendered. The shader reads the vector describing where the current vertex should be moved to from the average optical flow map. The shader then modifies the position of the vertex accordingly. The vertex is finally mapped with the colour from the original position in the texture inside the fragment shader.

4 OVERVIEW OF METRICS

Metrics and methods are used to measure frame per second (FPS) performance and output image quality improvement of the algorithms. The metrics and methods used are:

- Measurement of FPS performance;
- Image sharpness measurement;
- Aberration measurement; and
- Modulation Transfer Function (MTF) measurement.

4.1 FPS performance

The FPS performance of the algorithms is related to the algorithms complexity. The greater the complexity of the algorithm, the more resources and processing time it will require and will result in a lower FPS measurement.

4.2 Image sharpness

The image sharpness measurement uses the Laplacian Operator Sharpness Metric described in equation 5 to obtain a value indicative of the image sharpness. A larger value indicates a greater level of image sharpness.

4.3 Aberration

The aberration measurement measures the horizontal and vertical displacement present in a video, this relates to the geometric distortions due to atmospheric turbulence. The horizontal and vertical displacement lengths are recorded and compared to the other algorithms and unprocessed video footage.

4.4 Modulation Transfer Function (MTF)

The modulation transfer function across spatial frequencies is determined using:

$$MTF = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} / \frac{W - B}{W + B}, \tag{12}$$

Where:
I_{max} = the maximum intensity value within that frequency
I_{min} = the minimum intensity.
W = the maximum luminance for white areas
B = the minimum luminance for the dark areas.

A higher modulation index is more desirable because within a grey level image, the darkest and lightest grey areas will be further apart and details will be more pronounced.

5 EXPERIMENTAL SETUP

Each experiment required atmospheric turbulence affected video footage. Using a constructed imaging system as shown in figure 7, video footage was captured of target charts setup 1.2km away.

The constructed imaging system consisted of a telescope coupled to an internet protocol surveillance camera. The telescope became the ‘lens’ of the surveillance camera. The telescope used was a Celestron Nexstar 8 SE (8 inch Schmidt Cassegrain telescope), the large diameter allows a large amount of light to be collected, thus helping to reduce exposure times. An Arecont AV3100 surveillance camera was coupled to the telescope using a CS to 1.25 inch connector.

Each experiment made use of different target charts. In order to control the level of uncertainty in the results, all video footage was captured within one hour period to

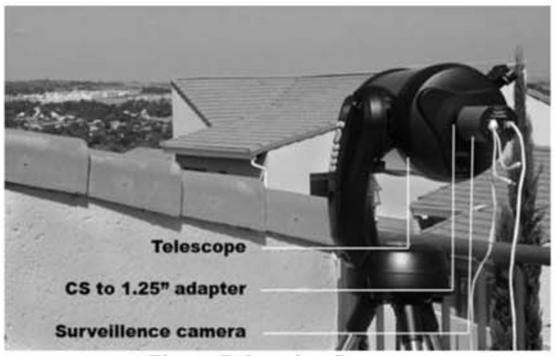

Figure 7: Imaging System

avoid dramatic changes in atmospheric conditions. Video footage was captured on a cloudless day to ensure consistent lighting conditions and shorter exposure times. Wind speed and direction was consistent over the period.

Each experiment made use of different target charts. In order to control the level of uncertainty in the results, all video footage was captured within one hour period to avoid dramatic changes in atmospheric conditions. Video footage was captured on a cloudless day to ensure consistent lighting conditions and shorter exposure times. Wind speed and direction was consistent over the period.

CPU	Intel Core 2 Duo E6750 @ 2.66 GHz
Motherboard	Asus P5K-Deluxe
RAM	2 GB DDR2 800 MHz

Table 1: Test PC Platform Specifications

5.1 *FPS performance*

The FPS performance of the algorithms is calculated using the time taken to process 100 image frames. This process was repeated 10 times for each video size and the average presented here. The same video footage is used for each of the algorithms so that results may be comparable.

5.2 *Image sharpness*

Sharpness measurements are taken across 50 image frames and an average calculated. Like the FPS performance experiment, the same video footage is processed by each of the algorithms and the sharpness measurement is then taken of the outputted videos so that results may be comparable.

5.3 *Aberration*

The aberration measurement is performed using a checkered chart of which video footage is captured through turbulent atmosphere. The video is then processed by the algorithms to determine their abilities to stabilize the footage. This method is similar to that used in [23]. The checkerboard target used consists of 20cm

in [23]. The checkerboard target used consists of 20cm
sized black and white squares. All the algorithms processed the same video footage to ensure exposure to and suppression of the same atmospheric turbulence conditions. After processing the video footage, maximum intensity values of the image are recorded for 50 image frames and then a grey level threshold is taken to identify square edges. The midpoint threshold value is determined from the maximum and minimum intensity values of the white and black blocks respectively. Horizontal and vertical lengths of the squares are measured and an average value calculated.

5.4 *Modulation Transfer Function (MTF)*

The MTF measurement experiment is similar to the method used in [24]. The spatial frequency charts used are viewed through atmospheric turbulence and the video footage captured and processed by each algorithm. An example of a spatial frequency chart containing various spatial frequency sizes is shown in Figure 8.

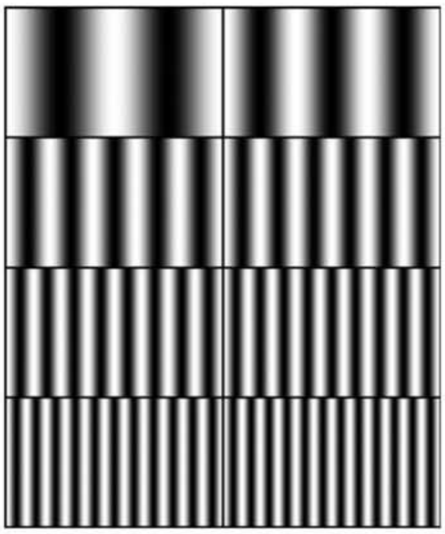

Figure 8: Spatial frequency chart example

Spatial frequencies used ranged from 415mm/lp (millimetre per line pair) to 10mm/lp. The captured video footage is processed by each of the algorithms and MTF measurements taken for each processed video using equation 12. The W and B intensity values are obtained from the lowest frequencies.

To account for the geometric distortions that vary across the video frames, a frame containing the maximum intensity values and one containing the minimum intensity values across 50 image frames are recorded. Measurements are taken from these two frames. More than 20 maximum intensity values are recorded and I_{max} for that frequency is calculated from the average. I_{min} is found in the same way for each spatial frequency except using the minimum intensity image that was constructed from the recordings.

6 RESULTS

For each of the video quality metrics human assessment was used to evaluate the effectiveness of the metric, and whether the metric was providing a pertinent insight into algorithm performance.

6.1 *FPS performance*

In this experiment a single 800x600 video sequence was resized to create a set of video sequences of various sizes. The CPU algorithms were run 10 times for each of the video sequences and the average frames processed per second was measured and recorded. The GPU implementation of the Lucas-Kanade based Wiener filtering and optical flow algorithm was executed on a number of different nVidia GPUs and one AMD/ATI GPU. The results of the experiment are presented in figure 9, a logarithmic scale had to be used to represent the results.

It can be seen that the CPU algorithms exhibit an exponential increase in processing time as the size of the videos increase exponentially. This was expected and demonstrates the serial nature of the CPU implementations. The Wiener filtering and Averaging algorithm performs the best out of the CPU implementations which was also expected, due to the fact that it is a much simpler algorithm than the optical flow based algorithms. The Lucas-Kanade based dewarping algorithm is faster than the Horn-Schunck based algorithm which is one of the reasons why it was chosen for the GPU implementation.

The GPU implementation exhibited very little change in processing time as the video sizes changed. This is due to the fact that the FFT and filtering steps of the algorithm need to use power-of-two sized textures and as such a texture of size 1024x1024 or 512x512 was used for the most intensive steps in the algorithm, and thus the processing times were very similar. The jump from the 1024x1024 textures to the 512x512 textures can be seen on the graph. This flat response is also due to the GPU's parallel nature allowing the GPU to scale with the increased video size much more effectively than in a serial architecture. At the largest resolutions the GPU implementations provide a performance increase of 3 orders of magnitude over the CPU algorithms.

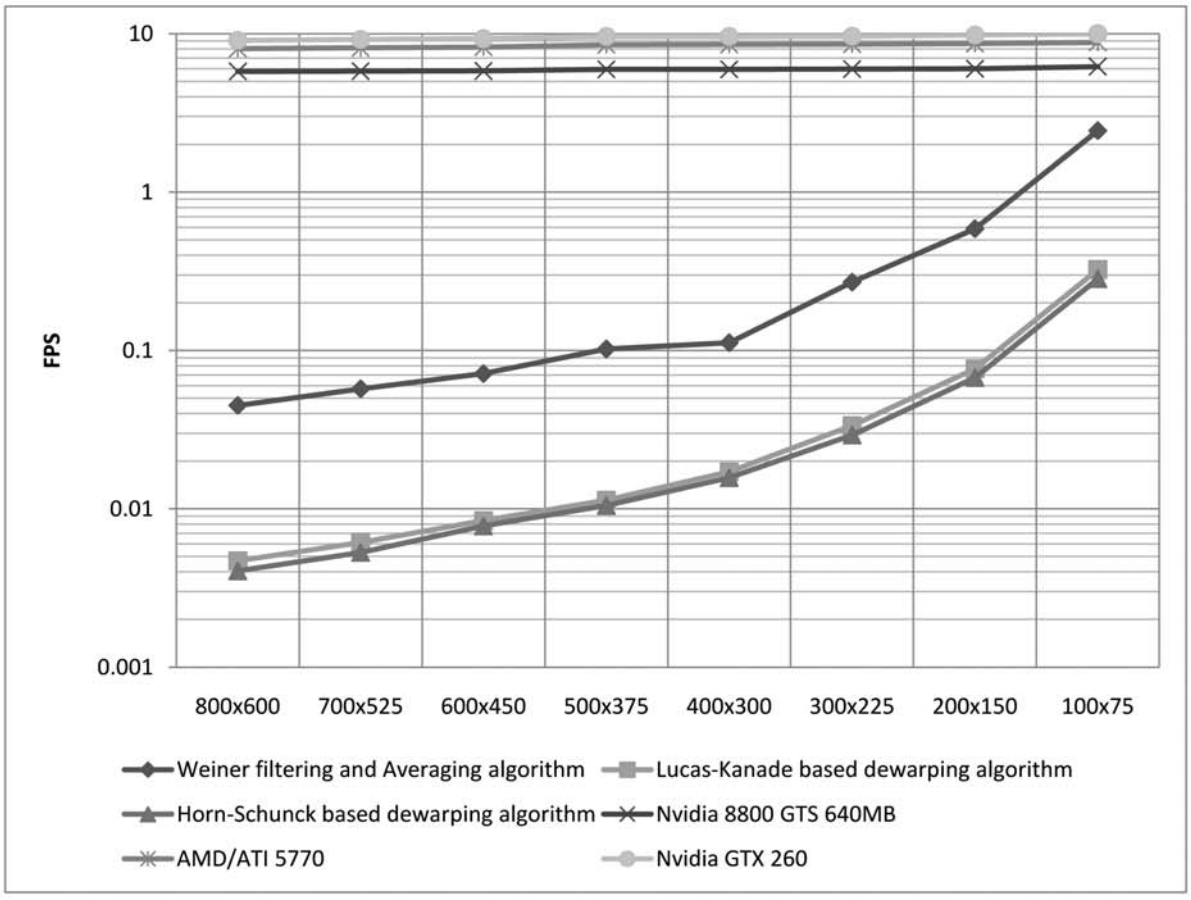

Figure 9: FPS Performance test results

6.2 Image sharpness

This experiment makes use of the Laplacian operator based sharpness measurement. A higher metric value indicates a sharper image.

Figure 10 shows the sharpness measurements for video processed using the Wiener filtering and Averaging algorithm making use of the shown range of ratios used in the ratio averaging portion of the algorithm. Also shown is the sharpness measurement for the unprocessed video

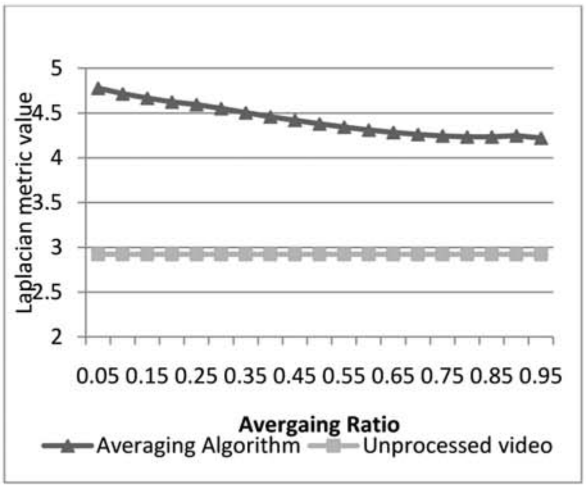

Figure 10: Wiener filtering and Averaging algorithm sharpness results

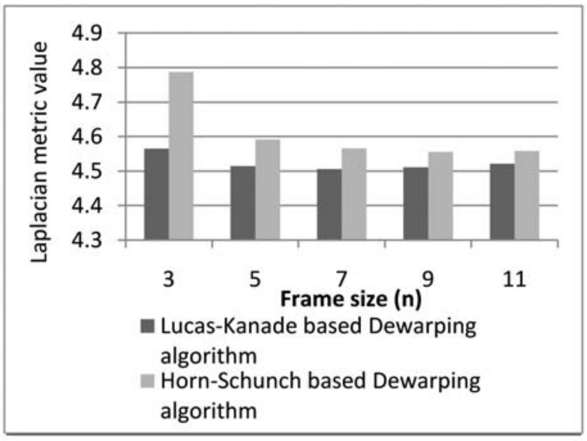

Figure 11: Optical flow based dewarping algorithm sharpness results

It is apparent that as the Averaging Ratio gets larger the sharpness of the resulting video decreases. This is due to the fact that the ratio dictates how much of the average frame is retained as each frame of the video is processed. It is apparent that the processed video is significantly sharper than the unprocessed video.

Figure 11 shows the sharpness measurements of the Lucas-Kanade based Dewarping algorithm and Horn-Schunck based dewarping algorithm. In this graph *n* indicates the size of frame being used to find the average optical flow that will be used to dewarp each frame. The smallest frame size results in the sharpest image as the dewarping operation occurs over fewer frames.

From figure 11 it is apparent that the Horn-Schunck based algorithm produces better results when using this metric. This however demonstrates a flaw with the metric being used as when the output videos are assessed by a human for both the Lucas-Kanade and Horn-Schunck based algorithms it is apparent that the Horn-Schunck video contains far more high frequency noise and is in fact less sharp. The sharpness metric measures the high frequency content in the frames and thus produces a strong response to this high frequency noise. This gives us these skewed results.

6.3 Aberration

The results of this experiment should illustrate the severity of the geometric aberrations present in the video sequence under consideration. The larger the displacements measured the more severe the aberrations in the video are.

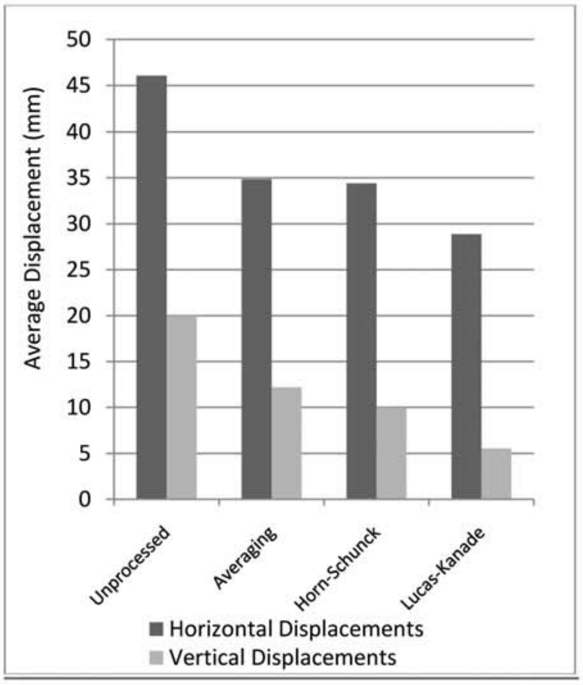

Figure 12: Displacement results

From figure 12 it can be seen that as expected the horizontal displacements are far higher than the vertical displacements due to the lateral effect of the wind. The results show that all the algorithms produce a reduction of the measured aberrations in the video and thus are stabilizing the geometric distortion present in the video. It can be seen from the results that the Lucas-Kanade based dewarping algorithm reduces the measured aberrations in the processed video the most dramatically out of the 3 algorithms. While the Averaging algorithm can potentially stabilize the video the most effectively out of the three algorithms this results in a blurring effect which can be detected by the aberration metric.

6.4 Modulation Transfer Function (MTF)

The blurring effect of the atmosphere suppresses the higher spatial frequencies that can be seen in an image. MTF analysis allows one to measure how much various spatial frequencies are degraded by the atmosphere. This determines the level of spatial detail that is visible in a given image or video.

The MTF (Modulation Transfer Function) provides the spatial frequency response of the video sequence in question. A higher response will mean that the specific spatial frequencies are more visible. From the figures it is apparent that the MTF curve of the unprocessed video exhibits a steeper gradient than the MTF curves of the processed video sequences.

In the lowest frequencies there is not a striking improvement for any of the three algorithms but it can be seen that the three algorithms do improve the MTF curve at these frequencies somewhat. The Horn-Schunck based algorithm performed the best at the lowest frequencies.

The frequencies that we are the most interested in are the high spatial frequencies, as these frequencies are the worst affected by blurring and contain the spatial detail we are trying to restore. It is apparent that these frequencies also show the biggest variation in results between the three algorithms.

The Wiener filtering and Averaging algorithm performs the worst out of the three algorithm and at times seems to worsen the MTF response when compared to the unprocessed video. This is due to the blurring effect of the averaging portion of the algorithm which will degrade the higher frequencies present in the video. It is clear that the Lucas-Kanade based algorithm has the shallowest curve and highest response at the higher frequencies. The Horn-Schunck based algorithm has the second best response and shows an increase in the MTF curve for the entire range of spatial frequencies.

6.5 Final remarks

From the various metrics it is apparent that the Lucas-Kanade based Dewarping algorithm appears to produce the sharpest and most stable video results of the three algorithms under consideration. It can also been seen that the Laplacian operator sharpness metric is over sensitive to high frequency noise and as such is not a useful measure of sharpness on its own.

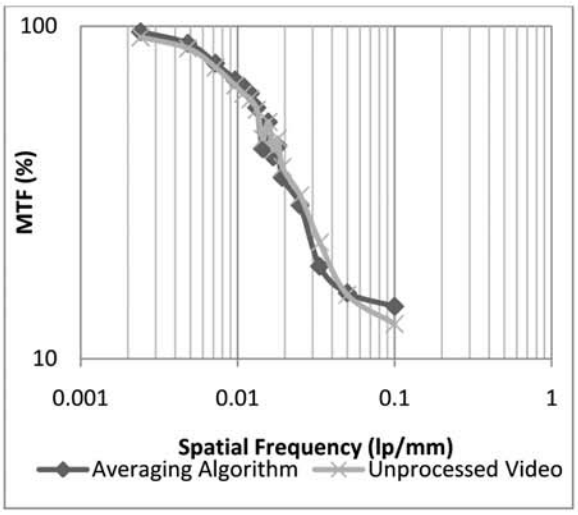

Figure 13: MTF Curve of Averaging Algorithm

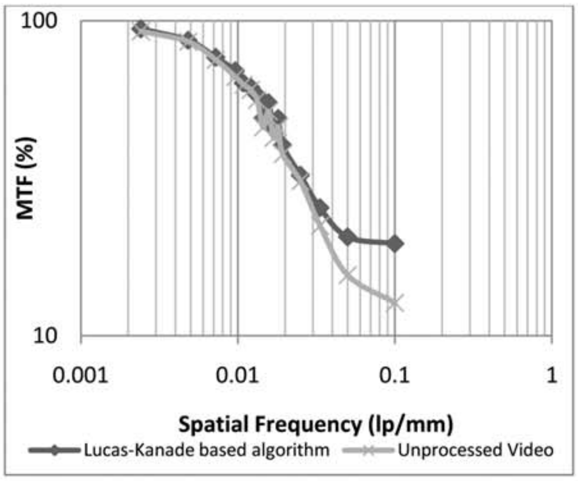

Figure 14: MTF curve of Lucas-Kanade based dewarping algorithm

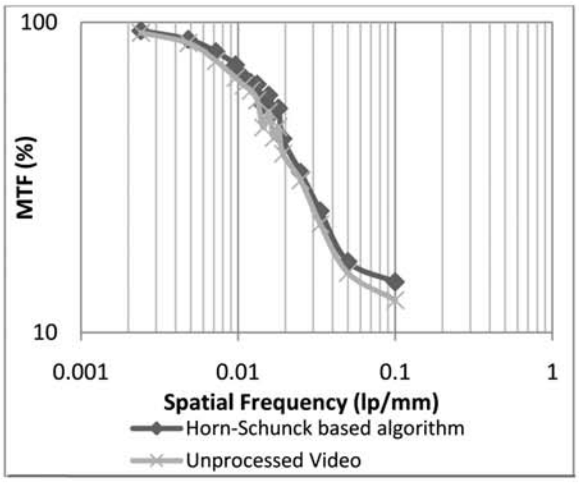

Figure 15: MTF curve of Horn-Schunck based dewarping algorithm

7 CONCLUSION

Long range surveillance systems have numerous advantages within the military and civil fields, however at long distances, the atmospheric turbulence caused by the heat of the earth's surface distorts the received image making object identification difficult. There are a number of approaches to mitigating the effect of the turbulent atmosphere and numerous documented algorithms in the literature. There are however no clear comparisons between algorithms and the metrics that could be used to compare the performance of these algorithms

In this paper three algorithms were implemented based on techniques described in the literature. A number of performance metrics were proposed and used to attempt to compare the performance of these algorithms. Human assessment was used to determine whether the metrics were providing realistic and pertinent information about the video scenes under consideration.

The performance metrics all seemed to provide valuable insight into the performance and natures of the algorithms. However it was found that the Laplacian sharpness metric was overly sensitive to high frequency noise and as such is not a useful metric when used in isolation.

Using the proposed metrics, we found that the Wiener filtering and Averaging algorithm does stabilize the geometric distortion present in the video, but it does so at the price of introducing further blurring. The Wiener filtering and Optical flow based methods seemed to provide stable and sharper results. Of these two algorithms the Lucas-Kanade based algorithm seemed to perform the best resulting in the most stable and sharp output out of the three algorithms.

The GPU implementation of the Lucas-Kanade based algorithm was fairly successful with only a few stages that required intensive redesign to allow for better exploitation of the parallel GPU architecture. The issue of recursion was easily solved using a ping pong approach and a vertex shader was used to implement the scattering operation used for dewarping.

The Averaging algorithm does, however, seem to have the lowest computational requirements of the three algorithms, but the GPU implementation of the Lucas-Kanade based algorithm did provide a significant speed increase and promises easy scalability to achieve real-time processing speeds in the future.

8 FUTURE WORK

The purpose of this study was to provide an entry point into the topic of mitigating the effects of atmospheric turbulence on video captured over a long range. As such there is much more possible work to be done to take the first steps presented in this study and produce a fully functional real world system to address the problem of heat shimmer.

A few specific areas that need improvement are as follows:

- Develop a more advanced dewarping scheme,
- Improve the accuracy of the optical flow methods,
- Develop a non-uniform deblurring scheme,
- Improve the GPU implementation.

9 REFERENCES

[1] W. Zhao, L. Bogoni, M. Hansen, "Video Enhancement by Scintillation Removal", *icme,* , 2001 IEEE International Conference on Multimedia and Expo (ICME'01), 2001, pp. 71.

[2] A. Bovik, Handbook of Image and Video Processing, Academic Press, San Diego, 2000.

[3] S.D. Ford[1], B.M. Welsh[1], M.C. Roggemann[2], "Constrained least-squares estimation in deconvolution from wave-front sensing", Optics communications, [1]Department of Electrical and Computing Engineering, Air Force Institute of Technology, 2950 P Street, Bldg 640, Wright-Patterson Air Force Base, OH 45433-7765, USA, [2]Department of Electrical Engineering, Michigan Technological University, 1400 Townsend Drive, 121 EERC bldg, Houghton, MI 49931-1295, USA. 1997.

[4] B.R. Frieden, "An exact, linear solution to the problem of imaging through turbulence", *Optical sciences centre,* University of Arizona, Tucson, AZ 85721, USA, 1997.

[5] G. Barnard. Restoration of Turbulence Degraded Images, Literature study report, Doc. No. QT-RA02-D-P047-001, Thales Advanced Engineering (PTY) LTD., October 1990.

[6] D.H. Frakes, J.W. Monaco, M.J.T. Smith, "Suppression of Atmospheric Turbulence in Video Using an Adaptive Control Grid Interpolation Approach", School of Electrical and Computing Engineering, Georgia Institute of Technology, Atlanta, GA, 2001.

[7] D. Li, R. Mersereau, D. H. Frakes, M. J. T. Smith, "New method for suppressing optical turbulence in video", Proc. EUSIPCO, 2005.

[8] G. Thorpe[1], A. Lambert[2], D. Fraser[2], "Atmospheric Turbulence Visualization through Image Time-Sequence Registration", [1]*Boeing Australia Limited,* 363 Adelaide Street, Brisbane, QLD 4000, Australia, [2]*School of Electrical Engineering,* University College, University of New South Wales, ADFA Canberra ACT 2600, Australia.

[9] B. Cong, "Encoding Neural Networks to Compute the Atmospheric Point Spread Function", Dept. of Computer Science, California State University, Fullerton, CA 92834 - USA.

[10] I. Goss-Ross, G. Barnard, B. Coetzer. *Restoration of Turbulence Degraded Images,* Algorithm

Development, Doc. No. RP-RA02-S-C047-003, Thales Advanced Engineering (PTY) LTD., March 1991.

[11] C. Max, "Introduction to Adaptive Optics and its History", American Astronomical Society 197th meeting, 2001.

[12] R. E. Hufnagel and N. R. Stanley, "Modulation transfer function associated with image transmission through turbulence media," Journal of the Optical Society of America A , vol. 54, pp. 52–61, 1964.

[13] E. C. Ifeacher, B. W. Jervis, Digital Signal Processing: A Practical Approach, Second Edition, Prentice Hall, Pearson Education Ltd. 2002.

[14] B.D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", Proceedings of Imaging understanding workshop, pp. 121-130, 1981.

[15] B.K.P. Horn, B.G. Schunck, "Determining optical flow", Artificial Intelligence, vol. 17, pp. 185-203, 1981.

[16] J.L. Barron, D.J. fleet, S.S. Beauchemin, T.A. Burkitt, "Performance of Optical Flow techniques", CVPR, 1992.

[17] D. Luebke, "The Democratization of Parallel Computing", SC07 Presentation, 2007.

[18] Lighthouse 3D, GLSL Tutorial, http://www.lighthouse3d.com/opengl/glsl/, September 2008.

[19] R.S. Wright Jr, B. Lipchak and N. Haemel, OpenGL SuperBible 4th Edition, Addison Wesley Professional, 2007.

[20] R.J. Rost, OpenGL Shading Language 2nd Edition, Addison Wesley Professional, 2006.

[21] D Göddeke, GPGPU: Basic Math Tutorial, http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/tutorial.html, September 2008.

[22] K. Moreland and E. Angel, "The FFT on a GPU", *SIGGRAPH/Eurographics Workshop on Graphics Hardware 2003 Proceedings*, pp. 112–119, 2003.

[23] B.D. Walters, "Comparison of two terrestrial atmospheric turbulence suppression algorithms", *IEEE Africon 2007*, ISBN: 978-1-4244-0987-7, pp. 1-7, Oct. 2007.

[24] C. J. Carrano, "Speckle imaging over horizontal paths", Proceedings of the SPIE —High Resolution Wavefront Control: Methods, Devices, and Applications IV, 4825, 2002.