Steerable Filters and Cepstral Analysis for Optical Flow Calculation from a Single Blurred Image

Ioannis M. Rekleitis
Center for Intelligent Machines, McGill University,
3480 University St., Montreal, Québec, Canada H3A 2A7,
e-mail: yiannis@cs.mcgill.ca

Abstract

This paper considers the explicit use of motion blur to compute the Optical Flow. In the past, many algorithms have been proposed for estimating the relative velocity from one or more images. The motion blur is generally considered an extra source of noise and is eliminated, or is assumed nonexistent. Unlike most of these approaches, it is feasible to estimate the Optical Flow map using only the information *encoded* in the motion blur. An algorithm that estimates the velocity vector of an image patch using the motion blur only is presented; all the required information comes from the frequency domain. The first step consists of using the response of a family of steerable filters applied on the *log* of the Power Spectrum in order to calculate the orientation of the velocity vector. The second step uses a technique called *Cepstral Analysis*. More precisely, the *log* power spectrum is treated as another signal and we examine the Inverse Fourier Transform of it in order to estimate the magnitude of the velocity vector. Experiments have been conducted on artificially blurred images and with real world data.[1]

1 Introduction

One of the fundamental problems in early Computer Vision is the measurement of motion in an image, frequently called optical flow. In many cases when a scene is observed by a camera there exists motion created either by the movement of the camera or by the independent movement of objects in the scene. In both cases, the goal is to assign a 3D velocity vector to each visible point in the scene; such an assignment is called the *velocity map*. In general it is impossible to infer from one view the 3D velocity map; however, most motion estimation algorithms calculate the projection of the velocity map onto the imaging surface. A large number of different algorithms have been developed in order to solve this problem.

The problem of estimating the optical flow has received much attention because of its many different applications. Tasks such as passive scene interpretation, image segmentation [14], surface structure reconstruction, inference of egomotion, and active navigation [11], [17], all use optical flow as input information.

Until now, most motion estimation algorithms considered optical flow with displacements of only a few pixels per frame. This approach limits the applications to slower motions and fails to seriously address the issue of motion blur, moreover, it works on images that are considered to be taken with infinitely small exposure time, more or less in a "stop and shoot" approach, which limits the real time applications.

The novel algorithm we have developed is based on interpreting the cue of *motion blur* to estimate the optical flow field in a *single image*. A key observation is that motion blur introduces a certain structure, a ripple, in the Fourier transform that can be detected and quantified using a modified form of *cepstral analysis*. Unlike classical approaches to visual motion analysis that rely upon operators tuned to specific spatial and temporal frequencies at specific orientations, our new approach makes use of all the information that can be gathered from a patch of the image and is thus quite robust [19].

The first step in our motion blur analysis is to compute the log power spectrum of a local image patch. Motion blur leads to a tell-tale ripple, centered at the origin, with orientation perpendicular to the orientation of the velocity vector. This orientation can be reliably determined, even in the presence of noise, using a steerable second Gaussian derivative filter. The magnitude of the velocity, which is related

[1]Appeared in "Vision Interface", pages 159-166, Toronto, May 1996.

to the period of the ripple, can then be determined by first collapsing the log spectrum data into a 1-D vector and then performing a second Fourier transform to yield the *cepstrum*, in which the magnitude of the velocity is clearly identified by a negative peak. The computational complexity of this algorithm is bounded by the Fast Fourier Transform operation, which is $O(n \log n)$, where n is the number of pixels in the image patch. Applying this analysis throughout the image provides an estimation of the complete optical flow field.

In most biological visual systems, the analysis of motion is critical; interesting experiments have been made with the visual system of the pigeon, rabbit, frog, fly, and more. The psychophysical aspects of motion information has been demonstrated by Ullman [20] and Marr [15]. During the last twenty years many algorithms have been proposed in order to calculate the optical flow. The first attempt comes from Horn and Schunck [12], [13], who used a differential approach. Since then many other algorithms have been proposed, which are generally divided into different categories according to the way they handled the data used to calculate the optical flow. Similar studies exist for biological as well as computer visual systems [20], [15]. The use of a series of linear filters has occurred in the past in order to solve questions about stereopsis, texture and optical flow from a set of images [22]. Also, research has been conducted in order to ensure the robustness of the results of optical flow calculation [5] and for solving the problem when partial information is known [2].

Section 2 of this paper presents the description of the problem, and the computational model for the motion blur. The extraction of the orientation of the motion from the frequency domain and different methods to improve the results appear in Section 3. Section 4 deals with *Cepstral analysis* and the extraction of the magnitude. Section 5 provides the results from the simulated and real world images. The summary and future goals are the subjects of Section 6.

2 Motion Blur

When a changing scene is observed by a camera, most of the existing algorithms assume that it is possible to take pictures every δt instantly, which means that every picture is taken with a $dt \approx 0$ exposure time. If that is not the case, then the exposure time $(dt = T)$ is large enough that different points in the scene are moving far enough and consequently their corresponding projections on the image plane travel several pixels. Therefore, during the capture of an image, at any single image point, a certain number of scene points is projected during the exposure time, each one contributing to the final brightness of the image point. It is clear that the blurring of the image exists only across the direction of the motion; this one dimensional blur is called *Motion Blur* (see figure 2a). Motion blur is of particular interest in the biological research also and many studies about its significance in the perception of the world have been done [3], [10], [6]. Earlier work in the estimation of the motion blur parameters has used different methods as the bispectrum [7], or the *Discrete Cosine Transform* [23]; in both cases the orientation of the motion was assumed known, assumption that was false in a lot of the applications.

Ideal motion blur can be described mathematically [8] as the result of a linear filter $\mathbf{b}(x,y) = \mathbf{i}(x,y) * \mathbf{h}(x,y)$ where $\mathbf{i}$ is the theoretical image taken with an exposure time $T_e = 0$, $\mathbf{b}$ is the real blurred image, and $\mathbf{h}$ the point spread function (PSF). Given an angle= α and the length $d = V_o \times T_e$, which is the number of scene points that affect a specific pixel, the point spread function of motion blur is zero everywhere except at a line segment with length d at an angle α with the x-axis, where it has the value $\frac{1}{d}$.

3 Optical Flow Calculation

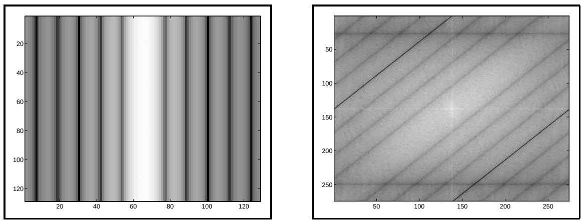
In order to calculate the optical flow for a certain point we make use of the area around it – this method needs only one frame taken with an exposure time δt where the motion blur spans for more than a couple of pixels, as is the situation in a series of applications. To estimate the Optical Flow map of the whole image we run the following described algorithm for a series of overlapping image segments. The algorithm can be divided in two stages: first there is the extraction of the orientation of the velocity vector from the Fourier Spectrum with the use of a set of Steerable filters, and second the calculation of the magnitude of it from the Cepstrum.

3.1 Spectral Analysis

An image blurred due to motion is usually represented by a linear system of a convolution: $g(x,y) = f(x,y) * h(x,y)$ with $h(x,y)$ the convolution kernel that cause the blur. In general, for an arbitrary direction of the motion the *FFT* of the *PSF* is a ripple as shown in figure 1, clear in the case of horizontal or vertical motion (see figure 1a) or distorted slightly[2]

[2]Mainly because of numerical errors and the windowing effect. We have to take into account also the fact that FT is a complex transformation and therefore it exist an imaginary

– as is the case for a blur at the 45° angle (see figure 1b) where it is more the shape of an ellipse with the long axis perpendicular to the direction of motion.



(a) (b)

Figure 1: The Power Spectrum of the *PSF* of horizontal (a) and at 45° angle (b) motion blur

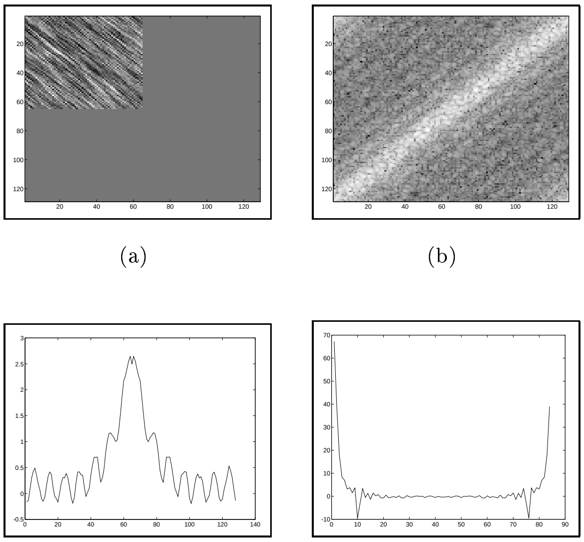
The Power Spectra of the blurred image is the product of the Power Spectra of the PSF multiplied by the Power Spectra of the unblurred image (see figure 2b). If the unblurred image is rich in texture then the main structure of the Power Spectra of the blurred image is the ripple that appear across the direction of the motion.

An important source of noise in the frequency domain comes from the ringing effect when we take only a part of the image, the more abrupt the change into the zero level of the masking window, the more severe the artifacts that are going to appear. Many masking function have been proposed up to now in order to minimize the ringing effect and at the same time to preserve the information existing in the image patch [18]. In this algorithm the Gaussian Masking function has been used. Also, In order to get a more (optically) detailed frequency image, we could add zeros at the end of the signal, in both dimensions, and then take the Fourier Transform (see figures 2a,b), this technique is called Zero Padding [18], and it increases the sampling rate of the *FT*.

3.2 Orientation Extraction: Steerable Filters

As we saw earlier, the Power Spectrum of the blurred image is characterised by a central ripple that goes across the direction of the motion. In order to extract this orientation we treat the Power Spectra as an image and a linear filter is applied so it could identify the orientation of the ripple. More specifically the second derivative of a two dimensional Gaussian

part that is not displayed here.



(a) (b)

(c) (d)

Figure 2: A zero padded image patch (a), its *Fourier Spectrum* (b), the *Fourier Spectrum* collapsed (c), and the *Cepstrum* (d).

is used. The second derivative of the Gaussian along the x-axis is $G_2^0 = \frac{\partial^2 G}{\partial x^2}$. If we filter the Power Spectrum of a blurred image with G_2^0 we are going to get maximum response when the ripple is across the x-axis. In order to extract the orientation of the ripple, we have to find the angle θ in which the filter of the second derivative of a Gaussian – oriented at that angle (G_2^θ) – is going to give the highest response. Fortunately, the second derivative of the Gaussian G_2^θ belongs to a family of filters called "steerable filters" [9], whose response can be calculate at any angle θ based only on the responses of three *basis filters*.

$$RG_2^\theta = k_a(\theta)RG_{2a} + k_b(\theta)RG_{2b} + k_c(\theta)RG_{2c} \quad (1)$$

The response of the second derivative of the Gaussian at an angle θ (RG_2^θ) is given in equation 1. The set of the three basis filters is shown in the left column of the table 1 and in the right column we could see the three *interpolation functions* that are used.

3.3 Cepstral Analysis

To improve robustness, the magnitude of the velocity is calculated using a 1D projection of the Power

$G_{2a} = 0.921(2x^2 - 1)e^{\mu}$	$k_a(\theta) = \cos^2(\theta)$
$G_{2b} = 1.843xye^{\mu}$	$k_b(\theta) = -2\cos(\theta)\sin(\theta)$
$G_{2c} = 0.921(2y^2 - 1)e^{\mu}$	$k_c(\theta) = \sin^2(\theta)$
$\mu = -(x^2 + y^2)$	

Table 1: The three *basis filters* and their *interpolation functions*

Spectra onto the line across the velocity vector orientation that passes through the origin.

If only one line of the blurred image is taken (across the direction of the motion) then the blurred signal is equivalent to the convolution of the unblurred signal by the step function which in the frequency domain is transformed into the sinc function ($sinc(x) = \frac{\sin x}{x}$). The period of the sinc pulse is equivalent to the length of the step function, which is in turn equivalent to the velocity magnitude. If we take the Fourier Transform of the sinc function, its period appears as a negative peak.

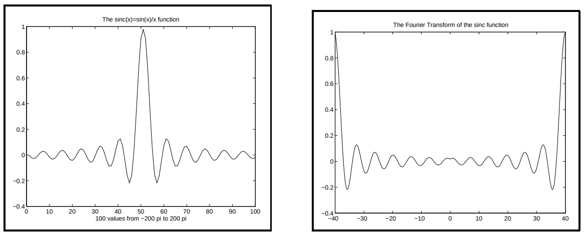
In order to approximate the 1D signal we collapse the Power Spectra from 2D into 1D. The resulting signal has also the shape of the sinc function, because the ripple caused by the motion blur is the dominant feature (see figures 3a and 2c). Every pixel P(x,y) in the Power Spectra is mapped into the line that passes through the origin O at an angle θ with the x-axis equal to the orientation of the motion, and at distance $d = x\cos(\theta) + y\sin(\theta)$. If we take the Fourier Transform of the sinc function we have an almost identical shape with the one that appears when we take the Fourier Transform of the collapsed spectrum (compare 3b and 2d).

3.3.1 Definitions

The 1D signal with the approximate shape of the sinc function is treated as a new signal and its Fourier Transform is calculated, this technique is called cepstral analysis.

The most common definition of the *Cepstrum* [3] of a function $f(x,y)$ is $\mathcal{C}ep\{f(x,y)\} = \mathcal{F}^{-1}\{\log(F(\omega,v))\}$, where $F(\omega,v)$ is the Fourier Transform of a function $f(x,y)$ [18],[16]. In other words, it is the *Inverse Fourier Transform* of the log-

[3] *Cepstrum* is a juxtaposition of letters for the word *Spectrum*



(a) (b)

Figure 3: The Graphical representation of the *sinc* function (a), the Fourier Transform of the *sinc* function (b).

arithm of the *Fourier Transform* of the signal. The *Cepstrum* is a complex function; if we want to have only the real part then instead of the $F(\omega,v)$ we take its magnitude ($|F(\omega,v)|+1$) (which is the case in this algorithm) as in equation 2.

$$\mathcal{C}ep\{f(x,y)\} = \mathcal{F}^{-1}\{\log(1 + |F(\omega,v)|)\} \quad (2)$$

3.3.2 Magnitude extraction

As we see in the previous sections we have transformed the logarithm of the Power Spectrum of the blurred image into an 1D signal. This new signal has approximately the shape of a sinc ripple – distortions exist due to noise, windowing effect, and the process of collapsing the signal itself. The real part of the Cepstrum is used in order to estimate the length of the ripple, which is in fact the magnitude of the velocity vector. The signal we have is an artificial average signal of the logarithm of the Power Spectrum of the image. This has the advantage that the features in the Power Spectrum that were there due to the unblurred image have been cancelled out, leaving as a prominent characteristic the effect of the motion blur. As the 2D signal is collapsed across the direction of the motion it simulates a motion blur created by uniform movement across the x-axis and has the appearance of the $sinc(x) = \frac{\sin x}{x}$,

4 Results

A series of experiments have been conducted using the above mentioned algorithm. An implementation in *Matlab* and C was used with two categories of input data. The first category consists of stationary images, natural or artificially created, that we artificially blur by simulating the results of motion blur;

the second category consists of real images taken by a camera with the existence of relative motion between the camera and the scene. The data from the first category give us the ability to check the validity of our results and perform error measurements, while the images from the second category are ensuring that the algorithm is working on real world data.

4.1 Simulation Data

Two images have been used in this section, each one of them having different properties. The first one (figure 4a) is a real image taken by a stationary camera, with many different features such as smooth surfaces, edges, and highly textured areas. The second one (figure 4b) is a random noise picture, rich in texture, having the same size as the previous one. As we discussed earlier the algorithm is more effective with images rich in texture and this is quite obvious in the results we get, where erroneous results appear mainly over smooth surfaces.

Both images have been blurred by convolving the unblurred image with the same kernel. The motion is assumed to be at a direction of $+125°$ angle with the x-axis and with a length of 13 pixels. In real world the blur is created before the digitisation, therefore the points that contribute to the final value of the pixel appear in a straight line. When we try to reproduce the same results in the discrete space at an arbitrary angle, we have similar results to that of aliasing in graphics. In order to avoid that, the convolution matrix is created by using the technique of antialiasing lines, where the pixels are weighted according to their distance to an "abstract" line.

In order to get better results and to eliminate the ringing effect, a Gaussian window is used for masking before we proceed into the velocity vector estimation. Also in all cases zero padding has been used. The middle needle map in both figures (4c, 4d) is created using a 64×64 window, and the last one (4e, 4f) using a 128×128.

In the first image (figure 4a) the optical flow is calculated with worse precision at the more uniform areas. The error measures for the middle map (4c) are $3.6°$ for the average absolute error in angle and 5.2 pixels in distance. For the third map (4e) where a large window was used the results are much more improved with the average absolute error for the orientation at $2.2°$ and the magnitude at 5.7 pixels. The second image is pure texture and the results are even better. The middle needle diagram (figure 4d) presents decreased error measures with the average absolute error in orientation $3.0°$ and the magnitude 4.1 pixels. Part of the error comes from the way the artificial blurring was implemented through antialiasing lines. For the last velocity map (figure 4f) where a larger window (128×128) has been used the average absolute error is really small, $3.0°$ for the orientation and 4.1 pixels in distance.

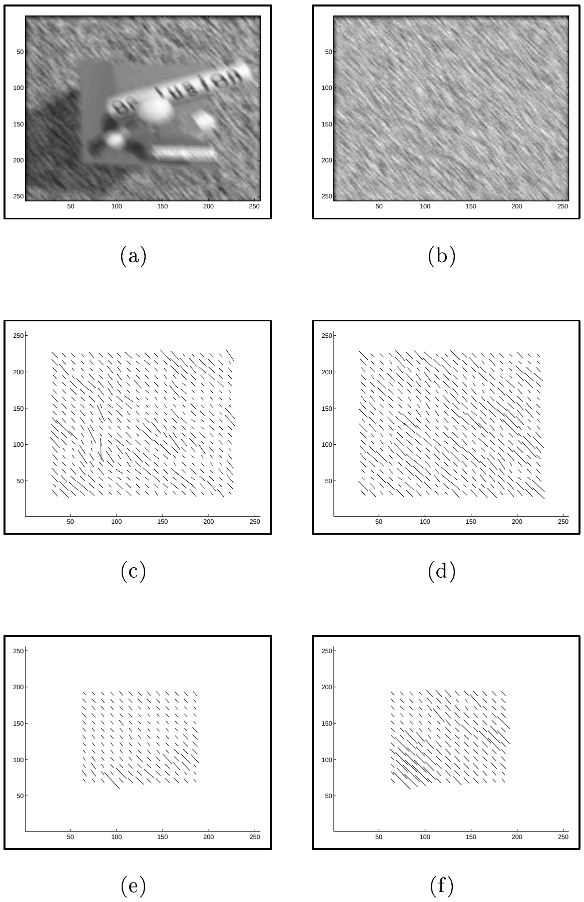
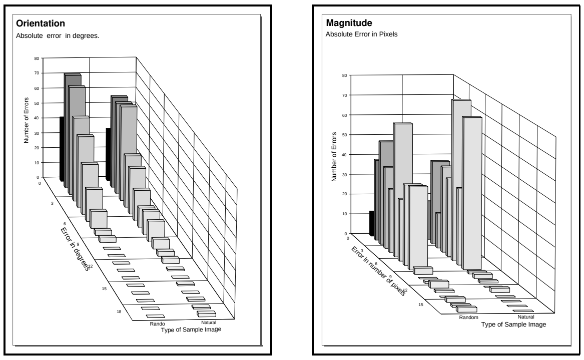


Figure 4: A natural image artificially blurred (a), a random noise image artificially blurred (b), the Optical flow Map of (a) and (b) respectivelly with a 64×64 window (c), (d) the Optical flow Map of (a) and (b) with an 128×128 window (e), (f).

An estimation on the distribution of the error can come from the error histograms presented in figure 5. The data come from the velocity maps of figures 4a and 4b. It is clear the importance of texture in the algorithm as the random noise image is better than the natural one. Another issue worth mentioning is the accuracy of the orientation estimation, where most of the results are accurate to two or three de-



(a) (b)

Figure 5: Error distribution of the velocity maps of figures 3b and 3b. Orientation absolute error (a), Magnitude absolute error (b).

grees.

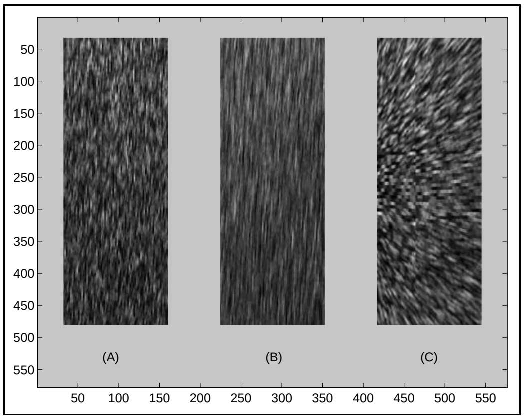
4.2 Real Data

The images in this case have been taken by a camera and immediately digitised into the computer. To achieve controlled motion between the camera and the scene the following setup was used in all except one cases: a camera was mounted on a base pointing downwards, and a plane (created by cardboard) with random dots on top of it was used as the main object in the scene. We moved the plane in different directions, with a speed high enough to produce motion blur with the preset exposure time of the camera. The format, for economy of space, consists of three different blurred images, labelled (A), (B), (C) in one figure, and their respectively Optical Flow maps in a second figure, following the same labelling. In all the experiments the same configurations have been used: we calculate the Optical Flow on a grid which is dense 10×10, using a 64×64 window. The patch of the blurred image was masked first with a Gaussian window (to avoid the ringing effect) and then zero padded up to 128×128.

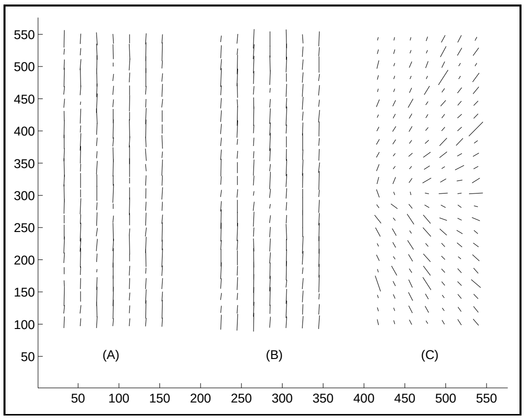
The first set of images is shown in figure 6a. The first image 6a(A) has been created by moving the plane in parallel with the y-axis with a steady and relatively small velocity; the algorithm has correctly estimated the orientation of the velocity almost everywhere, as can be seen in the Optical Flow map in figure 6b(A). The accuracy of the magnitude estimation is not clear, although if we compare it with the next image some qualitative results can be drawn. The second image, 6a(B), is created again with a steady velocity parallel to the y-axis, this time at a higher speed, fact that is easily noticeable by the length of the blur. Again the Optical Flow map, in figure 6b(B), has an accurate estimation of the orientation and also gives an average bigger magnitude for the velocity vectors. By comparing these two cases it is obvious that the orientation estimation is correct and also the magnitude estimation shows the difference between different speeds. The third image 6a(C) is created completely differently; the random-dot decorated plane is left to fall free under the camera and during that fall we take a snapshot. As can be seen from the blur lines, the focus of expansion is at the middle of the left side, and indeed the algorithm gives the same results. In the Optical Flow map (figure 6b(C)) we could see the velocity vectors pointing to the point of expansion and having a gradually decreasing magnitude as they reach that point.

In the next set of images two images were created by rotational motion, and one image was created with a completely different setup. The first image (figure 6c(A)) was created by moving the camera by hand horizontally across a self full of books and binders.[4] The lighting of the scene was low and therefore some of the features didn't appear; in addition it is quite notable the lack of texture in a lot of the areas. In spite of these problems the velocity vectors in majority have the correct orientation and approximately the same magnitude, (figure 6d(A)) results that agree with the blurred image. The last two images are created by rotating the random-dot plane under the camera, with different speeds. In the middle image (figure 6c(B)) the centre of rotation is in the upper right part and the speed is high. In figure 6d(B) we could see the velocity vectors having the proper orientation, and a rather big magnitude. The last instance, 6c(C), is taken with the plane considerably close to the camera and with a smaller rotation speed; the centre of rotation is in the upper left corner, where the pixels are rather discrete. A smooth Optical Flow map is presented in figure 6d(C) with the vectors having the correct orientation, circular around the upper left corner in the location of the centre of rotation, and having an almost constant magnitude.

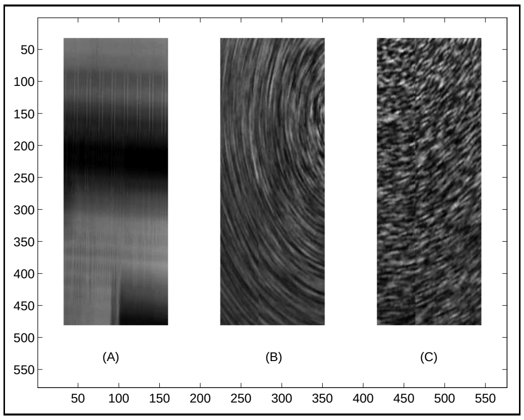
[4] The image is rotated by 90° due to the way *Matlab* is handling the images; taking that into account, the spiral binding of some of the books is quite obvious.



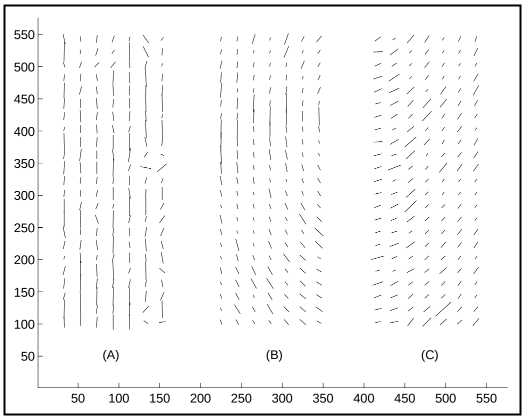
(a)



(b)



(c)



(d)

Figure 6: (a),(c)Three Images with motion blur ,The Optical Flow map of (a) and (c) using a 64×64 window with a step of 20 pixels, with zero padding and Gaussian masking is presented at (b) and (d)

5 Conclusions

In this paper a new approach calculating the optical flow map using motion blur is formulated and evaluated experimentally. An algorithm is presented for computing the *optical flow* from a single motion-blurred image, using only the information present in the structure imposed on the image by the motion blur.

The algorithm can be considered as operating in two steps. For each patch of the image the direction of motion is first determined and then the speed in that direction is recovered. The algorithm operates in the frequency domain, where it exploits the fact that motion blur introduces a characteristic *ripple* in the power spectrum. The orientation of these ripples in the 2D power spectrum is perpendicular to the direction of the motion blur. A key element of the algorithm developed in this thesis is the robust and efficient identification of the orientation of these ripples by making use of *steerable filters*. In the experimental results, the orientation of motion blur is often recovered to within just a few degrees.

Once an accurate estimate of the orientation of the motion blur is known, the speed of motion, or the spatial extent of the blur, can be computed using a modified form of *cepstral analysis*. The first step in this procedure is to collapse the 2D log power spectrum into a 1D signal along the line indicating the direction of motion. The frequency of the ripple in the resulting 1D signal can be identified by taking a further Fourier Transform and locating a negative peak.

There are some limitations for the applicability of this algorithm that are worth noting. Most importantly the algorithm depends on the presence of texture in the image, since the blur in a region with homogeneous brightness is undetectable. The magnitude of motion blur that can be detected is limited by the size of the image patch being analyzed. Also, if the motion blur is too small, on the order of just a few pixels, it becomes indistinguishable from other small-scale features, such as texture, noise, or out-of-focus blur.

This algorithm has been implemented and evaluated experimentally using artificial and natural images. The results acquired are very promising: the orientation of the velocity vector is accurately estimated ($1°$ to $3°$ average error), and the magnitude calculations are satisfied for qualitative estimations. Our algorithm has the advantage of exploiting information in a motion-blurred image that traditional motion analysis methods have tended to ignore. It has also the added advantage of providing an optical flow map from a single image, instead of a sequence of images. The algorithm also lends itself easily to efficient parallel implementation.

References

[1] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images-a review. *Proceedings of the IEEE*, 76(8):917–935, August 1988.

[2] Nicola Ancona and Tomaso Poggio. Optical flow from 1d correlation. pages 209–214. IEEE, 1993.

[3] Charles H. Anderson. Blur into focus. *Nature*, 343:419–420, February 1990.

[4] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal on Computer Vision*, 12(1):43–77, 1994.

[5] Michael J. Black and P. Anandan. A framework for the robust estimation of optical flow. pages 231–236. IEEE, 1993.

[6] C. Bonnet. Visual motion detection models:features and frequency. *Perception*, 6:491–500, 1977.

[7] Michael M. Chang, Murat A. Tekalp, and Tanju A. Erdem. Blur identification using the bispectrum. *IEEE Transactions on Signal Processing*, 39(10):2323–2325, Octomber 1991.

[8] R. Fabian and D. Malah. Robust identification of motion and out-of-focus blur parameters from blurred and noisy images. *CVGIP: Graphical Models and Image Processing*, 53(5):403–412, September 1991.

[9] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

[10] Thomas L. Harrington and Marcia K. Harrington. Perception of motion using blur pattern information in the moderate and high-velocity domains of vision. *Acta Psychologica*, 48:227–237, 1981.

[11] Martin Herman and Tsai-Hong Hong. Visual navigation using optical flow. In *Proc. NATO Defense Research Group Seminar on Robotics in the Battlefield*, pages 1–9. NATO, Paris France, March 1991.

[12] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. Technical report, Massachusetts Institute of Technology, 1980.

[13] Berthold Klaus Paul Horn. *Robot Vision*. MIT Press, McGraw-Hill, 1986.

[14] H. A. Mallot, H. H. Bulthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64:177–185, 1991.

[15] D. Marr. *Vision*. Freeman, New York, 1982.

[16] William K. Pratt. *Digital Image Processing*. John Wiley & Sons, Inc., 1978.

[17] K. Prazdny. Egomotion and relative depth map from optical flow. *Biological Cybernetics*, 36:87–102, 1980.

[18] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing*. Macmillan Publishing Company, 866 Third Avenue, New York, New York 10022, second edition, 1992.

[19] Ioannis M. Rekleitis. Visual motion estimation based on motion blur interpretation. Master's thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, 1995.

[20] Shimon Ullman. The interpetation of visual motion. Technical report, Massachusetts Institute of Technology, 1979.

[21] J. F. Vega-Riveros and K. Jabbour. Review of motion analysis techniques. *IEE Proceedings*, 136(6):397–404, December 1989.

[22] Joseph Weber and Jitendra Malik. Robust computation of optical flow in a multi-scale differential framework. In *International Conference in Computer Vision*, pages 12–20, 1993.

[23] Yasuo Yoshida, Kazuyochi Horiike, and Kazuhiro Fujita. Parameter estimation of uniform image blur using dct. *IEICE Trans. Fundamentals*, E76(7):1154–1157, July 1993.