A Motion Detection Algorithm Using Local Phase Information

Aurel A. Lazar Nikul H. Ukani
Yiyin Zhou*
Department of Electrical Engineering,
Columbia University, New York, NY 10027

October 14, 2015

Abstract

Previous research demonstrated that *global* phase alone can be used to faithfully represent visual scenes. Here we provide a reconstruction algorithm of visual scenes by only using *local* phase information. We also demonstrate that local phase alone can be effectively used to detect local motion. The local phase-based motion detector is akin to models employed to detect motion in biological vision, e.g., the Reichardt detector.

The local phase-based motion detection algorithm introduced here consists of two building blocks. The first building block measures/evaluates the temporal change of the local phase. The temporal derivative of the local phase is shown to exhibit the structure of a second order Volterra kernel with two normalized inputs. We provide an efficient, FFT-based algorithm for implementing the change of the local phase. The second processing building block implements the detector. The local phase-based motion detection algorithm compares the maximum of the Radon transform of the local phase derivative with a chosen threshold.

We demonstrate examples of applying the local phase-based motion detection algorithm on several video sequences. We also show how the locally detected motion can be used for segmenting moving objects in video scenes and compare our local phase-based algorithm to segmentation achieved with a widely used optic flow algorithm. Our results suggest that local spatial phase information may provide an efficient alternative to perform many visual tasks *in silico* as well as *in vivo* biological vision systems.

Keywords: phase-based motion detection, bio-inspired motion detection, reconstruction of

*The author's names are alphabetically listed.

Correspondence: Aurel A. Lazar,
Department of Electrical Engineering,
Columbia University,
500 West 120th Street,
New York, NY 10027, United States
Email: aurel@ee.columbia.edu (AAL), nikul@ee.columbia.edu (NHU), yiyin@ee.columbia.edu (YZ)

visual scenes from phase, FFT, Radon Transform.

Contents

1 Introduction

Following Marr, the design of an information processing system can be approached on multiple levels [1]. Figure 1 illustrates two levels of abstraction, namely, the algorithmic level and the physical circuit level. On the algorithmic level, one studies procedurally how the information is processed independently of the physical realization. The circuit level concerns the actual realization of the algorithm in physical hardware, *e.g.*, a biological neural circuit or silicon circuits in a digital signal processor. In this paper, we put forth a simple motion detection algorithm that is inspired by motion detection models of biological visual systems (in vivo neural circuit) and provide an efficient realization that can easily be implemented on commodity (in silico) DSP chips.

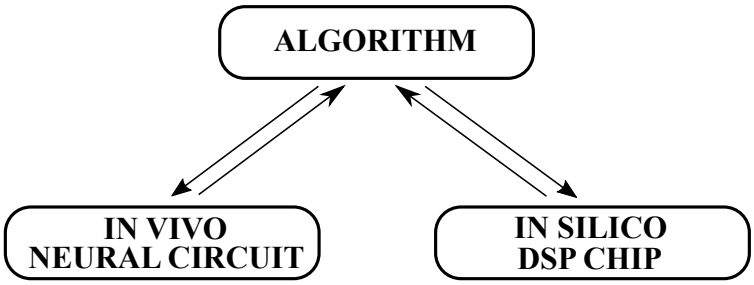


Figure 1: Algorithms can have different physical realizations. For example, an algorithm can be implemented by neural circuits in a biological system. Alternately, it can be implemented on a digital signal processor. Algorithms direct the implementation on the physical layer. Conversely, biological neural circuits inspire new algorithmic designs, which can, in turn, be expressed and improved upon by a realization *in silico*.

Visual motion detection is critical to the survival of animals. Many biological visual systems have evolved highly efficient/effective neural circuits to detect visual motion. Motion detection is performed in parallel with other visual coding circuits and starts already in the early stages of visual processing. In the retina of vertebrates, it is known that at least three types of Direction-Selective Ganglion Cells (DSGC) are responsible for signaling visual motion at this early stage [2]. In flies, direction-selective neurons are found in the optic lobe, 3 synapses away from the photoreceptors [3].

The small number of synapses between photoreceptors and direction-selective neurons suggests that the processing involved in motion detection is not highly complex but still very effective. In addition, the biological motion detection circuits are organized in a highly parallel way to enable fast, concurrent computation of motion. It is also interesting to note that the early stages of motion detection are carried out largely in the absence of spiking neurons, indicating that initial stages of motion detection are preferably performed in the "analog" domain. Taking advantage of continuous time processing may be critical for quickly processing motion since motion intrinsically elicits fast and large changes in the intensity levels, that is, large amounts of data under stringent time constraints.

Modern, computer-based motion detection algorithms often employ optic flow techniques to estimate spatial changes in consecutive image frames [4, 5]. Although, often time, optic flow

estimation algorithms produce accurate results, the computational demand to perform many of these algorithms is too high for real-time implementation.

Several models for biological motion detection are available and their architecture is quite simple [6]. The Reichardt motion detector [7] was thought to be the underlying model for motion detection in insects [8]. The model is based on a correlation method to extract motion induced by spatio-temporal information patterns of light intensity. Therefore, it relies on a correlation/multiplication operation. A second model is the motion energy detector [9]. It uses spatio-temporal separable filters and a squaring nonlinearity to compute motion energy and it was shown to be equivalent to the Reichardt motion detector. Earlier work in the rabbit retina was the foundation to the Barlow-Levick model [10] of motion detection. The model relies on inhibition to compensate motion in the null direction.

In this paper, we provide an alternative motion detection algorithm based on local phase information of the visual scene. Similar to mechanisms in other biological models, it operates in continuous time and in parallel. Moreover, the motion detection algorithm we propose can be efficiently implemented on parallel hardware. This is, again, similar to the properties of biological motion detection systems. Rather then focusing on velocity of motion, we focus on localization, *i.e.*, where the motion occurs in the visual field as well as the direction of motion.

It has been shown that images can be represented by their global phase alone [11]. Here we provide a reconstruction algorithm of visual scenes by only using *local* phase information, thereby demonstrating the spectrum of the representational capability of phase information. The Fourier shift property clearly suggests the relationship between the global shift of an image and the global phase shift in the frequency domain. We elevate this relationship by computing the change of local phase to indicate motion that appears locally in the visual scene. The local phases are computed using window functions that tile the visual field with overlapping segments, making it amenable for a highly parallel implementation. In addition, we propose a Radon-transform-based motion detection index on the change of local phases for the readout of the relation between local phases and motion.

Interestingly, phase information has been largely ignored in the field of linear signal processing and for good reason. Phase-based processing is intrinsically non-linear. Recent researches, however, showed that phase information can be smartly employed in speech processing [12] and visual processing [13]. For example, spatial phase in an image is indicative of local features such as edges when considering phase congruency [14]. The role of spatial phase in computational and biological vision, emergence of visual illusions and pattern recognition is discussed in [15]. Together with our result for motion detection, these studies suggest that phase information has a great potential for achieving efficient visual signal processing.

This paper is organized as follows. In Section 2, we show that local phase information can be used to faithfully represent visual information in the absence of amplitude information. In Section 3, we develop a simple local phase-based algorithm to detect motion in visual scenes and provide an efficient way for its implementation. We then provide examples and applications of the proposed motion detection algorithm to motion segmentation. We also

compare our results to those obtained using motion detection algorithms based on optic flow techniques in Section 4. Finally, we summarize our results in Section 5.

2 Representation of Visual Scenes Using Phase Information

The use of complex valued transforms, is widespread, for both representing and processing images. When represented in polar coordinates, the output of a complex valued transform of a signal can be split into amplitude and phase. In this section, we define two types of phases of an image: global phase and local phase. We then argue that both types of phases can faithfully represent an image and we provide a reconstruction algorithm that recovers the image from local phase information. This indicates that phase information alone can largely represent an image or video signal.

It will become clear in the sections that follow that the use of phase for representing of image and video information leads to efficient ways to implement certain types of image/video processing algorithms, e.g., motion detection algorithms.

2.1 The Global Phase of Images

Recall that the Fourier transform of a real valued image $u = u(x, y), (x, y) \in \mathbb{R}^2$, is given by

$$\hat{U}(\omega_x, \omega_y) = \int_{\mathbb{R}^2} u(x, y) e^{-j(\omega_x x + \omega_y y)} dx dy \tag{1}$$

with $\hat{U}(\omega_x, \omega_y) \in \mathbb{C}$ and $(\omega_x, \omega_y) \in \mathbb{R}^2$.

In polar coordinates, the Fourier transform of u can be expressed as

$$\hat{U}(\omega_x, \omega_y) = \hat{A}(\omega_x, \omega_y) e^{j\hat{\phi}(\omega_x, \omega_y)}, \tag{2}$$

where $\hat{A}(\omega_x, \omega_y) \in \mathbb{R}$ is the amplitude and $\hat{\phi}(\omega_x, \omega_y)$ is the phase of the Fourier transform of u.

Definition 1. *The amplitude of the Fourier transform of an image $u = u(x, y), (x, y) \in \mathbb{R}^2$, is called the global amplitude of u. The phase of the Fourier transform of an image $u = u(x, y), (x, y) \in \mathbb{R}^2$, is called the global phase of u.*

It is known that the global phase of an image plays an important role in the representation of natural images [11]. A classic example is to take two images and to exchange their global phases before their reconstruction using the inverse Fourier transform. The resulting images are slightly smeared but largely reflect the information contained in the global phase.

2.2 The Local Phase of Images

The global phase indicates the offset of sinusoids of different frequencies contained in the entire image. However, it is not intuitive to relate the global phase to local image features such as edges and their position in an image. To study these local features, it is necessary to modify the Fourier transform such that it reflects properties of a restricted region of an image. It is natural to consider the Short-Time (-Space) Fourier Transform (STFT)

$$U(\omega_x, \omega_y, x_0, y_0) = \int_{\mathbb{R}^2} u(x,y) w(x - x_0, y - y_0) e^{-j(\omega_x(x-x_0)+\omega_y(y-y_0))} dxdy, \tag{3}$$

where $w = w(x,y), (x,y) \in \mathbb{R}^2$, is a real valued window function centered at $(x_0, y_0) \in \mathbb{R}^2$. Typical choices of window functions include the Hann window and the Gaussian window. The (effectively) finite support of the window restricts the Fourier transform to local image analysis.

Similarly to the Fourier transform, the STFT can be expressed in polar coordinates as

$$U(\omega_x, \omega_y, x_0, y_0) = A(\omega_x, \omega_y, x_0, y_0) e^{j\phi(\omega_x, \omega_y, x_0, y_0)}, \tag{4}$$

where $A(\omega_x, \omega_y, x_0, y_0) \in \mathbb{R}$ is the amplitude and $\phi(\omega_x, \omega_y, x_0, y_0)$ is the phase of the STFT.
Definition 2. *The amplitude of the STFT of an image $u = u(x,y), (x,y) \in \mathbb{R}^2$, is called the local amplitude of u. The phase of the STFT of an image $u = u(x,y), (x,y) \in \mathbb{R}^2$, is called the local phase of u.*

Note that when w is a Gaussian window, the STFT of u evaluated at $(\omega_x, \omega_y, x_0, y_0)$ can be equivalently viewed as the response of a complex-valued Gabor receptive field

$$h(x,y) = e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} e^{-j(\omega_x(x-x_0)+\omega_y(y-y_0))} \tag{5}$$

to u. In this case, the window of the STFT clearly is given by

$$w(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{6}$$

Therefore, the STFT can be realized by an ensemble of Gabor receptive fields that are common in modeling simple and complex cells (neurons) in the primary visual cortex [16].

2.3 Reconstruction of Images from Local Phase

Amplitude and phase can be interpreted as measurements/projections of images that are indicative of their information content. Classically, when both the global amplitude and phase are known, it is straightforward to reconstruct the image. The reconstruction calls for computing the inverse Fourier transform given the global amplitude and phase. Similarly, when using local amplitude and phase, if the sampling functions form a basis or frame in a space of images, the reconstruction is provided by the formalism of wavelet theory, *e.g.*, using Gabor wavelets [17].

Amplitude or phase represent partial information extracted from visual scenes. They are obtained via nonlinear sampling, *i.e.*, a nonlinear operation for extracting the amplitude and phase information from images. The nonlinear operation makes reconstruction from either amplitude or phase alone difficult. Earlier studies and recent development in solving quadratic constraints, however, suggest that it is possible to reconstruct images from global or local amplitude information [18, 19].

While computing the amplitude requires a second order (quadratic) nonlinearity, computing the phase calls for higher order nonlinear operators (Volterra kernels). It is possible, however, to reconstruct up to a constant scale an image from its global phase information alone without explicitly using the amplitude information. An algorithm was provided for solving this problem in the discrete signal processing domain in [11]. The algorithm smartly avoids using the inverse tangent function by reformulating the phase measurements as a set of linear equations that are easy to solve.

Using a similar argument, we demonstrate in the following that, up to a constant scale, a bandlimited signal $u = u(x, y), (x, y) \in \mathbb{R}^2$, can be reconstructed from its local phase alone. We first formulate the encoding of an image u by local phase, using the Gabor receptive fields as a special case. It is straightforward then to formulate the problem with local phase computed from other types of STFTs.

Formally, we consider an image, $u = u(x, y)$, on the domain $\mathbb{R}^2$, to be an element of a space of trigonometric polynomials $\mathcal{H}$ of the form

$$u(x,y) = \sum_{l_x=-L_x}^{L_x} \sum_{l_y=-L_y}^{L_y} c_{l_x l_y} e_{l_x l_y}(x,y), \tag{7}$$

where

$$e_{l_x l_y} = \exp\left(j l_x \frac{\Omega_x}{L_x} x\right) \exp\left(j l_y \frac{\Omega_y}{L_y} y\right), l_x = -L_x, \cdots, L_x, l_y = -L_y, \cdots, L_y, \tag{8}$$

are the set of basis functions of $\mathcal{H}$, and Ω_x, L_x are the bandwidth and the order, respectively, of the space in the x dimension and Ω_y, L_y are the bandwidth and the order, respectively, of the space in the y dimension.

$\mathcal{H}$ is a Reproducing Kernel Hilbert Space with inner product [20], [21]

$$\langle u_1, u_2 \rangle = \int_{[0,T_x]\times[0,T_y]} u_1(x,y)\overline{u_2(x,y)} dxdy, \tag{9}$$

where $T_x = \frac{2\pi L_x}{\Omega_x}, T_y = \frac{2\pi L_y}{\Omega_y}$ are the period of the space in the x and y dimensions, respectively.

Consider a bank of N Gabor receptive fields

$$h_{kl,mn}(x,y) = \mathcal{T}_{kl}(w(x,y)e^{-j(\omega_{x_m}x + \omega_{y_n}y)}), \tag{10}$$

where $\mathcal{T}_{kl}$ is the translation operator with $\mathcal{T}_{kl}u = u(x - kb_0, y - lb_0)$, $k, l \in \mathbb{Z}$, $b_0 > 0$, $0 \leq kb_0 \leq T_x$, $0 \leq lb_0 \leq T_y$, and $\omega_{x_m} = m\omega_0, \omega_{y_n} = n\omega_0$, $m, n \in \mathbb{Z}$, $\omega_0 > 0$, $-\Omega_x \leq m\omega_0 \leq \Omega_x$,

$-\Omega_y \leq n\omega_0 \leq \Omega_y$. The responses of the Gabor receptive fields to the input $u(x,y)$ are given by

$$\begin{aligned}\int_{\mathbb{R}^2} u(x,y)h_{kl,mn}(x,y)dxdy &= \int_{\mathbb{R}^2} u(x,y)w(x-kb_0,y-lb_0)e^{-j(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0))}dxdy \\ &= A_{kl,mn}e^{j\phi_{kl,mn}},\end{aligned} \tag{11}$$

where $A_{kl,mn} \geq 0, A_{kl,mn} \in \mathbb{R}$, is the local amplitude and $\phi_{kl,mn} \in [0, 2\pi)$ is the local phase.

Dividing both sides of (11) by $e^{j\phi_{kl,mn}}$, we have

$$\int_{\mathbb{R}^2} u(x,y)w(x-kb_0,y-lb_0)e^{-j(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn})}dxdy = A_{kl,mn}. \tag{12}$$

Since $A_{kl,mn} \in \mathbb{R}$, we have

$$\int_{\mathbb{R}^2} u(x,y)w(x-kb_0,y-lb_0)\cos(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn})dxdy = A_{kl,mn} \tag{13}$$

and

$$\int_{\mathbb{R}^2} u(x,y)w(x-kb_0,y-lb_0)\sin(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn})dxdy = 0. \tag{14}$$

Note that $w(x-kb_0,y-lb_0)\sin(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn})$ is a real-valued Gabor receptive field with a preferred phase at $\phi_{kl,mn}+\frac{\pi}{2}-\omega_{x_m}kb_0-\omega_{y_n}lb_0$.

Remark 1. *Assuming that the local phase information $\phi_{kl,mn}$ is obtained via measurements, i.e., filtering the image u with pairs of Gabor receptive fields* (10)*, the set of linear equations* (14) *has a simple interpretation: the image u is orthogonal to the space spanned by the functions*

$$w(x-kb_0,y-lb_0)\sin(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn}), \tag{15}$$

where $(k,l,m,n) \in \mathbb{I}$ with $\mathbb{I} = \{(k,l,m,n) \in \mathbb{Z}^4 | 0 \leq kb_0 \leq T_x,\ 0 \leq lb_0 \leq T_y,\ -\Omega_x \leq m\omega_0 \leq \Omega_x,\ -\Omega_y \leq n\omega_0 \leq \Omega_y\}$.

We are now in the position to provide a reconstruction algorithm of the image from phase $\phi_{kl,mn}$, $(k,l,m,n) \in \mathbb{I}$.

Lemma 1. *u can be reconstructed from $\phi_{kl,mn}$, $(k,l,m,n) \in \mathbb{I}$ as*

$$u(x,y) = \sum_{l_x=-L_x}^{L_x} \sum_{l_y=-L_y}^{L_y} c_{l_xl_y}e_{l_xl_y}(x,y), \tag{16}$$

with

$$\mathbf{\Phi c} = \mathbf{0}, \tag{17}$$

where $\mathbf{\Phi}$ is a matrix whose pth row and qth column entry is

$$[\mathbf{\Phi}]_{pq} = \int_{\mathbb{R}^2} w(x-kb_0,y-lb_0)\sin(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn})e_{l_xl_y}(x,y)dxdy. \tag{18}$$

Here p traverses the set $\mathbb{I}$, and $q = (2L_y+1)(l_x+L_x)+(l_y+L_y+1)$. $\mathbf{c}$ *is a vector of the form*

$$\mathbf{c} = \big[c_{-L_x,-L_y}, c_{-L_x,-L_y+1} \cdots, c_{-L_x,L_y}, c_{-L_x+1,-L_y}, c_{-L_x+1,-L_y+1}, \cdots, c_{-L_x+1,L_y}, \\ \cdots, c_{L_x,-L_y}, c_{L_x,-L_y+1}, \cdots, c_{L_x,L_y}\big]^T$$

that belongs to the null space of $\mathbf{\Phi}$*. A necessary condition for perfect reconstruction of* u*, up to a constant scale, is that,* $N \geq (2L_x+1)(2L_y+1)-1$*, where* N *is the number of phase measurements.*

Proof: Substituting (7) into (14), we obtain

$$\sum_{l_x=-L_x}^{L_x} \sum_{l_y=-L_y}^{L_y} c_{l_x l_y} \int_{\mathbb{R}^2} w(x-kb_0, y-lb_0) \sin(\omega_{x_m}(x-kb_0)+\omega_{y_n}(y-lb_0)+\phi_{kl,mn}) e_{l_x l_y}(x,y) dxdy = 0,$$

for all $k, l, m, n \in \mathbb{Z}$. Therefore, we have

$$\mathbf{\Phi c} = \mathbf{0},$$

inferring that $\mathbf{c}$ is in the null space of $\mathbf{\Phi}$.

If $N < (2L_x+1)(2L_y+1)-1$, it follows from the rank-nullity theorem that $dim(\text{null}(\mathbf{\Phi})) > 1$ leading to multiple linearly independent solutions to $\mathbf{\Phi c} = \mathbf{0}$. □

Example 1. *In Figure 2 an example of reconstruction of an image is shown using only local phase information. The reconstructed signal was scaled to match the original signal. The SNR of the reconstruction is* 39.97 *[dB]. An alternative way to obtain a unique reconstruction is to include an additional measurement, for example, the mean value of the signal* $\int_{\mathbb{R}^2} u(x,y)dxdy$ *to the system of linear equations* (14)*.*

Figure 2: An example of reconstruction of image from local phase information. (left) Original image, the dimension of the space is $3,969$. (middle) Reconstructed image from $4,896$ phase measurements, scaled to match the original. (right) Error.

3 Visual Motion Detection from Phase Information

In this section we consider visual fields that change as a function of time. For notational simplicity u will denote here the space-time intensity of the visual field.

3.1 The Global Phase Equation for Translational Motion

Let $u = u(x, y, t), (x, y) \in \mathbb{R}^2, t \in \mathbb{R}$, be a visual stimulus. If the visual stimulus is a pure translation of the signal at $u(x, y, 0)$, *i.e.*,

$$u(x, y, t) = u(x - s_x(t), y - s_y(t), 0), \tag{19}$$

where

$$s_x(t) = \int_0^t v_x(s)ds, s_y(t) = \int_0^t v_y(s)ds \tag{20}$$

are the total length of translation at time t in each dimension and, $v_x(t)$ and $v_y(t)$ are the corresponding instantaneous velocity components, then, the only difference between $u(x, y, t)$ and $u(x, y, 0)$ in the Fourier domain is captured by their global phase. More formally,

Lemma 2. *The change (derivative) of the global phase is given by*

$$\frac{d\hat{\phi}(\omega_x, \omega_y, t)}{dt} = -\omega_x v_x(t) - \omega_y v_y(t) = -[\omega_x, \omega_y][v_x(t), v_y(t)]^T, \tag{21}$$

where, by abuse of notation, $\hat{\phi}(\omega_x, \omega_y, t)$ denotes the global phase of $u(x, y, t)$, and $\hat{\phi}(\omega_x, \omega_y, 0)$ is the initial condition.

Proof: If $(\mathcal{F}u(\cdot, \cdot, 0))(\omega_x, \omega_y)$ is the 2D (spatial) Fourier transform of $u = u(x, y, 0), (x, y) \in \mathbb{R}^2$, by the Fourier shift theorem, we have

$$(\mathcal{F}u(\cdot, \cdot, t))(\omega_x, \omega_y) = (\mathcal{F}u(\cdot, \cdot, 0))(\omega_x, \omega_y)e^{-j(\omega_x s_x(t) + \omega_y s_y(t))}. \tag{22}$$

For a certain frequency component (ω_x, ω_y), the change of its global phase over time amounts to

$$\frac{d\hat{\phi}(\omega_x, \omega_y, t)}{dt} = -\omega_x \frac{ds_x(t)}{dt} - \omega_y \frac{ds_y(t)}{dt} = -\omega_x v_x(t) - \omega_y v_y(t) = -[\omega_x, \omega_y][v_x(t), v_y(t)]^T. \tag{23}$$

Therefore, in the simple case where the entire visual field is shifting, the derivative of the phase of Fourier components indicates motion, and it can be obtained by the inner product between the component frequency and the velocity vector. □

3.2 The Change of Local Phase

3.2.1 The Local Phase Equation for Translational Motion

The analysis in Section 3.1 applies to *global motion*. This type of motion occurs most frequently when the imaging device, either an eye or a camera, moves. Visual motion in the natural environment, however, is more diverse across the screen since it is, often time,

produced by multiple moving objects. The objects can be small and the motion more localized in the visual field.

Taking the global phase of u will not simply reveal where motion of independent objects takes places or their direction/velocity of motion. The ease of interpretation of motion by using the Fourier transform, however, motivates us to reuse the same concept in detecting local motion. This can be achieved, by restricting the domain of the visual field where the Fourier transform is applied.

To be able to detect local motion, we consider the local phase of $u(x, y, t)$ by taking the STFT with window function $w(x, y)$. Note that, the STFT and its ubiquitous implementation in DSP chips can be extensively used in any dimension. For simplicity and without loss of generality, we consider the window to be centered at $(0, 0)$. The STFT is given by

$$\int_{\mathbb{R}^2} u(x, y, t)w(x, y)e^{-j(\omega_x x+\omega_y y)}dxdy = A_{00}(\omega_x, \omega_y, t)e^{j\phi_{00}(\omega_x,\omega_y,t)}, \tag{24}$$

where, by abuse of notation, $A_{00}(\omega_x, \omega_y, t)$ is the amplitude, and $\phi_{00}(\omega_x, \omega_y, t)$ the phase.

Before we move on to the mathematical analysis, we can intuitively explain the relation between the change in local phase and visual motion taking place across the window support. First, if the stimulus undergoes a uniform change of intensity or it changes proportionally over time due to lighting conditions, for example, the local phase does not change since the phase is invariant w.r.t. intensity scaling. Therefore, the local phase does not change for such non-motion stimuli. Second, a rigid edge moving across the window support will induce a phase change.

For a strictly translational signal within the window support (footprint), *e.g.*,

$$u(x, y, t) = u(x - s_x(t), y - s_y(t), 0), \text{ for } (x, y) \in \text{supp}(w(x, y)), \tag{25}$$

where $s_x(t)$ and $s_y(t)$ are as defined in (20), we have the following result.

Lemma 3.

$$\frac{d\phi_{00}}{dt}(\omega_x, \omega_y, t) = -\frac{ds_x(t)}{dt}\omega_x - \frac{ds_y(t)}{dt}\omega_y + \mathfrak{v}_{00}(\omega_x, \omega_y, t), \tag{26}$$

where, by abuse of notation, $\phi_{00}(\omega_x, \omega_y, t)$ is the local phase of $u(x, y, t)$, and $\phi_{00}(\omega_x, \omega_y, 0)$ is the initial condition. The functional form of the term $\mathfrak{v}_{00} = \mathfrak{v}_{00}(\omega_x, \omega_y, t)$ is provided in Appendix A.

Proof: The derivation of Equation (26) and the functional form of $\mathfrak{v}_{00}$ is given in Appendix A. □

Remark 2. *We notice that the derivative of the local phase has similar structure to that of the global phase, but for the added term $\mathfrak{v}_{00}$. Through simulations, we observed that the first two terms in Equation* (26) *dominate over the last term for an ON or OFF moving edge [3]. For example, Figure 3 shows the derivative of the local phase given in equation* (26) *for an ON edge moving with velocity* $(40, 0)$ *pixels/sec.*

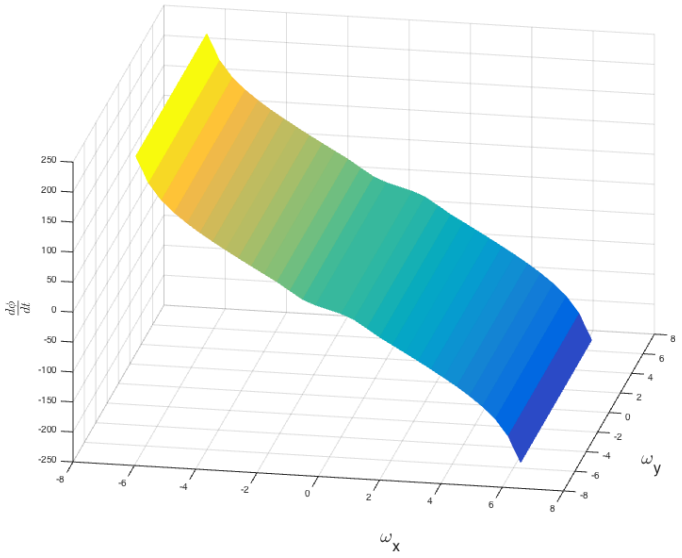


Figure 3: Derivative of the local phase for an ON edge moving in the positive x direction.

Remark 3. *Note that $\phi_{00}(\omega_x, \omega_y, t)$ may not be differentiable even if $u(x, y, t)$ is differentiable, particularly when $A_{00}(\omega_x, \omega_y, t) = 0$. For example, the spatial phase can jump from a positive value to zero when $A_{00}(\omega_x, \omega_y, t)$ diminishes. This also suggests that the instantaneous local spatial phase is less informative about a region of a visual scene whenever $A_{00}(\omega_x, \omega_y, t)$ is close to zero. Nevertheless, the time derivative of the local phase can be approximated by applying a highpass filter to $\phi_{00}(\omega_x, \omega_y, t)$.*

3.2.2 The Block Structure for Computing the Local Phase

We construct Gaussian windows along the x, y dimensions. The Gaussian windows are defined as

$$(\mathcal{T}_{kl} w)(x, y) = e^{-\frac{(x-x_k)^2+(x-y_l)^2}{2\sigma^2}}, \tag{27}$$

where $x_k = kb_0, y_l = lb_0$, $b_0 \in \mathbb{Z}^+$ is the distance between two neighboring windows and $1 \leq kb_0 \leq P_x$, $1 \leq lb_0 \leq P_y$, where $P_x, P_y \in \mathbb{Z}^+$ are the number of pixels of the screen in x and y directions, respectively.

We then take the 2D Fourier transform of the windowed video signal $u(x, y, t)(\mathcal{T}_{kl} w)(x, y)$ and write in polar form

$$\int_{\mathbb{R}^2} u(x, y, t)(\mathcal{T}_{kl} w)(x, y) e^{-j(\omega_x(x-x_k)+\omega_y(y-y_l))} dxdy = A_{kl}(\omega_x, \omega_y, t) e^{j\phi_{kl}(\omega_x, \omega_y, t)}. \tag{28}$$

The above integral can be very efficiently evaluated using the 2D FFT in discrete domain defined on $M \times M$ blocks approximating the footprint of the Gaussian windows. For example, the standard deviation of the Gaussian windows we use in the examples in Section 4 is 4 pixels. A block of 32×32 pixels ($M = 32$) is sufficient to cover the effective support (or

footprint) of the Gaussian window. At the same time, the size of the block is a power of 2, which is most suitable for FFT-based computation. The processing of each block (k, l) is independent of all other blocks; thereby, parallelism is readily achieved.

Note that the size of the window is informed by the size of the objects one is interested in locating. Measurements of the local phase using smaller window functions are less robust to noise. Larger windows would enhance object motion detection if the object size is comparable to the window size. However, there would be an increased likelihood of independent movement of multiple objects within the same window, which is not modeled here, and thereby may not be robustly detected.

Therefore, for each block (k, l), we obtain M^2 measurements of the phase $\phi_{kl}(\omega_{x_m}, \omega_{y_m}, t)$ at every time instant t, with $(\omega_{x_m}, \omega_{y_m}) \in \mathbb{D}^2$ where

$$\mathbb{D}^2 = \{(\omega_{x_m} = m\omega_0, \omega_{y_n} = n\omega_0), m, n = -M/2, -M/2 + 1, \cdots, M/2 - 1\},$$

with $\omega_0 = \frac{2\pi}{M}$. We then compute the temporal derivative of the phase, i.e., $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t)$ for $(\omega_x, \omega_y) \in \mathbb{D}^2$.

We further illustrate an example of the block structure in Figure 4. Figure 4a shows an example of an image of 64×64 pixels. Four Gaussian windows are shown each with a standard deviation of 4 pixels. The distance between the centers of two neighboring Gaussian windows is 6 pixels. The red solid square shows a 32×32-pixel block with $k = 3, l = 3$, which encloses effective support of the Gaussian window on top-left ($k = 0, l = 0$ is the block with Gaussian window centered at pixel $(1, 1)$). The green dashed square shows another 32×32-pixel block with $k = 3, l = 7$. The two Gaussian windows on the right are associated with the blocks $k = 7, l = 3$ and $k = 7, l = 7$, respectively. Cross section of all Gaussian windows with $k = 7, l \in [0, 10]$, that is, those centered on the magenta line, are shown in Figure 4b. The red and blue curve in Figure 4b correspond to the two Gaussian windows shown in Figure 4a. Figure 4b also suggests that some of the Gaussian windows are cut off on the boundaries. This is, however, equivalent to assuming that the pixel values outside the boundary are always zero, and it will not significantly affect motion detection based on the change of local phase.

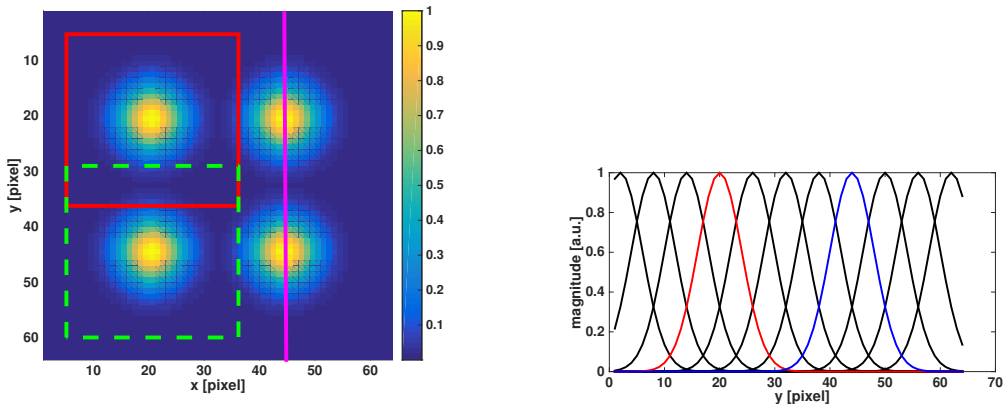
Since the phase, and thereby the phase change, is noisier when the local amplitude is low, an additional denoising step can be employed to discount the measurements of $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t)$ for low amplitude values $A_{kl}(\omega_x, \omega_y, t)$. The denoising is given by

$$\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t) \frac{A_{kl}(\omega_x, \omega_y, t)}{C(t) + \epsilon}, \tag{29}$$

where $\epsilon > 0$ is a constant, $C(t) = \frac{1}{M^2} \sum_{(\omega_x, \omega_y) \in \mathbb{D}^2} A_{kl}(\omega_x, \omega_y, t)$, and $(\omega_x, \omega_y) \in \mathbb{D}^2$.

3.3 The Phase-Based Detector

We propose here a block FFT based algorithm to detect motion using phase information. Such an algorithm is, due to its simplicity and parallelism, highly suitable for an *in silico* implementation.



(a) (b)

Figure 4: Example of block structure. (a) 4 yellow disks show 4 Gaussian window functions with translation parameter $k = 3, l = 3$ (top-left), $k = 3, l = 7$ (bottom-left), $k = 7, l = 3$ (top-right) and $k = 7, l = 7$ (bottom-right). The red solid square shows a 32×32-pixel block approximating the Gaussian window with $k = 3, l = 3$, and the green dashed square shows a 32×32-pixel block approximating the Gaussian window with $k = 3, l = 7$. (b) Cross section of all 11 Gaussian windows with translation parameters $k = 7, l \in [0, 10]$. The cross section is taken as indicated by the magenta line in (a), and the red and blue curve correspond to cross sections of the two Gaussian windows shown in (a) centered on the magenta line.

3.3.1 Radon Transform on the Change of Phases

We exploit the approximately linear structure of the phase derivative for blocks exhibiting motion by computing the Radon transform of $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t)$ over a circular bounded domain $C = \{(\omega_x, \omega_y) | (\omega_x, \omega_y) \in \mathbb{D}^2, \omega_x^2 + \omega_y^2 < \pi^2\}$.

The Radon transform of the change of phase in the domain C is given by

$$(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho, \theta, t) = \int_{\mathbb{R}} \frac{d\phi_{kl}}{dt}(\rho \cdot \cos\theta - s \cdot \sin\theta, \rho \cdot \sin\theta + s \cdot \cos\theta, t) \mathbb{1}_C(\rho \cdot \cos\theta - s \cdot \sin\theta, \rho \cdot \sin\theta + s \cdot \cos\theta) ds, \tag{30}$$

where

$$\mathbb{1}_C(\omega_x, \omega_y) = \begin{cases} 1 & \text{if } (\omega_x, \omega_y) \in C \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

The Radon transform $(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho, \theta, t)$ evaluated at a particular point (ρ_0, θ_0, t_0) is essentially an integral on $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)$ along a line oriented at angle $\frac{\pi}{2} + \theta_0$ with the ω_x axis and at distance $|\rho_0|$ along the $(\cos(\theta_0), \sin(\theta_0))$ direction from $(0, 0)$.

If for a particular k and l, we have $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t) = -v_x(t)\omega_x - v_y(t)\omega_y$, we have

$$\frac{(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho, \theta, t)}{\mathfrak{c}(\rho, \theta)} = \rho\Big[- v_x(t) \cos\theta - v_y(t) \sin\theta \Big], \tag{32}$$

where

$$\mathfrak{c}(\rho,\theta) = \int_{\mathbb{R}} \mathbb{1}_C(\rho\cdot\cos\theta - s\cdot\sin\theta, \rho\cdot\sin\theta + s\cdot\cos\theta)\ \ ds.$$

$\mathfrak{c}(\rho,\theta)$ is a correction term due to different length of line integrals for different (ρ,θ) in the bounded domain C.

After computing the Radon transform of $\frac{d\phi_{kl}}{dt}(\omega_x,\omega_y,t)$ for every block (k,l) at time t_0, we compute the Phase Motion Indicator (PMI), defined as

$$PMI_{kl} = \max_{\theta\in[0,\pi)} \sum_{\rho} \left| \frac{(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho,\theta,t_0)}{\mathfrak{c}(\rho,\theta)} \right|. \tag{33}$$

If the PMI_{kl} is larger than a chosen threshold, motion is deemed to occur in block (k,l) at time t_0.

Using the Radon transform makes it easier to separate rigid motion from noise. Since the phase is quite sensitive to noise, particularly when the amplitude is very small, the change of phase under noise may have comparable magnitude to that due to motion as mentioned earlier. The change of phase under noise, however, does not possess the structure suggested by Equation (26) in the (ω_x,ω_y) domain. Instead, it appears to be more randomly distributed. Consequently, the PMI value is comparatively small for these blocks (see also Section 3.3.3).

Moreover, the direction of the motion, for block (k,l) where motion is detected, can be easily computed as

$$\hat{\theta}_{kl} = \pi\left(\frac{\operatorname{sign}(\sum_{\rho>0} \frac{(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho,\alpha_{kl},t_0)}{\mathfrak{c}(\rho,\alpha_{kl})}) + 1}{2}\right) + \alpha_{kl}, \tag{34}$$

where

$$\alpha_{kl} = \operatorname*{argmax}_{\theta\in[0,\pi)} \sum_{\rho} \left| \frac{(\mathcal{R}\frac{d\phi_{kl}}{dt})(\rho,\theta,t_0)}{\mathfrak{c}(\rho,\theta)} \right|. \tag{35}$$

This follows from Equation (32).

3.3.2 The Phase-Based Motion Detection Algorithm

We formally summarize the above analysis as Algorithm 1 below. Figure 5 shows a schematic diagram of proposed phase-based motion detection algorithm.

The algorithm is subdivided into two parts. The first part computes local phase changes and the second part is the phase-based motion detector.

In the first part, the screen is divided into overlapping blocks. For example, the red, green, blue blocks in the plane "DIVIDE INTO OVERLAPPING BLOCKS" corresponds to the squares of the same color covering the video stream. A Gaussian window is then applied

on each block, followed by a 2D FFT operation that is used to extract the local phase. A temporal high-pass filter is then employed to extract phase changes.

In the second part, the PMI is evaluated for each block based on the Radon transform of the local phase changes in each block. Motion is detected for blocks with PMI larger than a preset threshold, and the direction of motion is computed as in (34).

It is easy to notice that the algorithm can be highly parallelized.

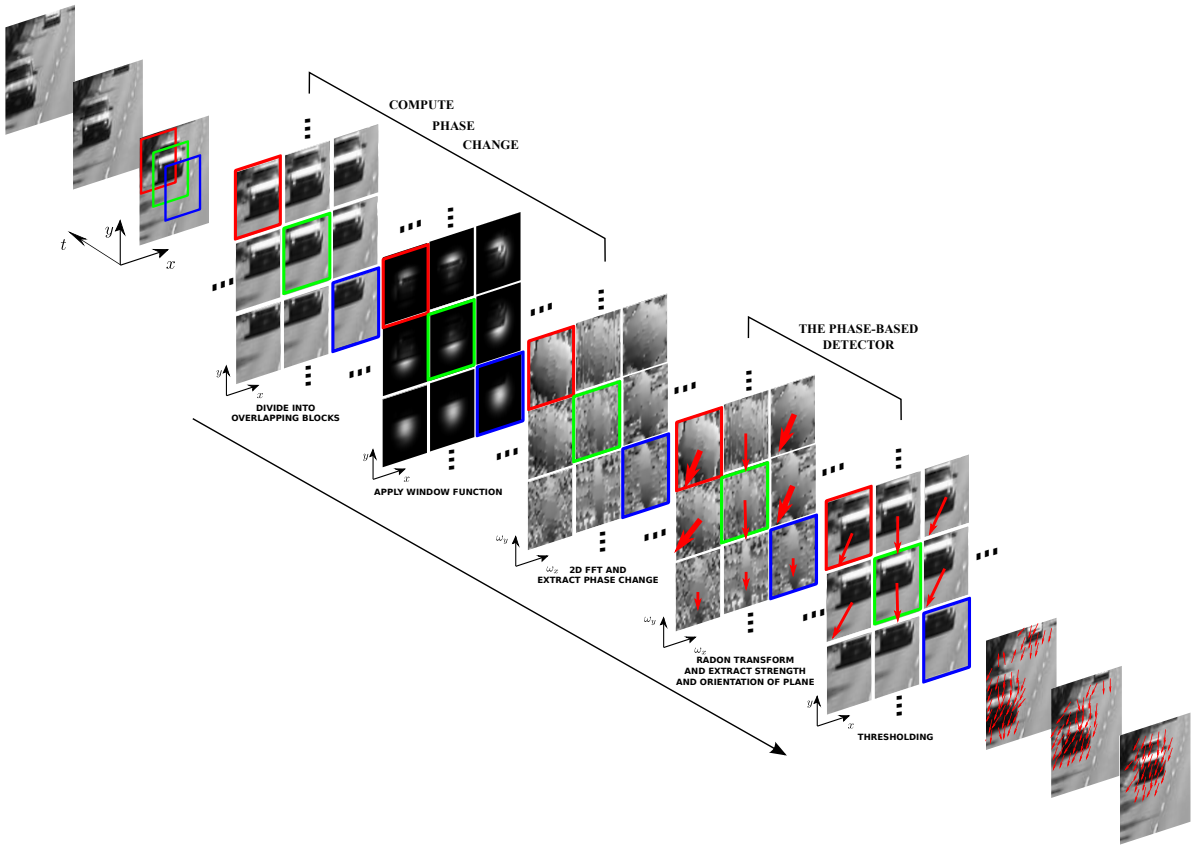


Figure 5: Schematic diagram of proposed phase-based motion detection algorithm.

3.3.3 Example

We provide an illustrative example in Figure 6 showing how motion is detected using Algorithm 1. The full video of this example can be found in Appendix **??**. Figure 6(a) depicts a still from the "highway video" in the ChangeDetection2014 dataset [22] evaluated at a particular time t_0. As suggested by the algorithm, the screen in Figure 6(a) is divided into 26×19 overlapping blocks and the window functions are applied to each block. Local phases can then be extracted from the 2D FFT of each windowed block, and the local phase changes are obtained by temporal high-pass filtering. The phase change is shown in Figure 6(b) for all blocks, with block $(12, 11)$ enlarged in Figure 6(c) and block $(23, 6)$ enlarged in Figure 6(d) (see also the plane "2D FFT AND EXTRACT PHASE CHANGE" in Figure 5). Note that

Input: Visual stream or Video data $u(x, y, t)$
Output: Detected Motion
Construct Gaussian window of size $M \times M$, denote as w;
for $t = \Delta T, 2\Delta T, \cdots, N\Delta T$ **do**
  Divide the screen of frame at time t into multiple $M \times M$ blocks $u_{kl}(t)$, with overlaps;
  foreach *block* $u_{kl}(t)$ **do**
    Multiply pixel-wise the block $u_{kl}(t)$ with the Gaussian window $u_{kl}(t)w$;
    Take $M \times M$ 2D FFT of $u_{kl}(t)w$, denote phase as $\phi_{kl}(\omega_x, \omega_y, t)$ and amplitude as $A_{kl}(\omega_x, \omega_y, t)$;
    foreach *frequency* (ω_x, ω_y) **do**
      Highpass filter, temporally, $\phi_{kl}(\omega_x, \omega_y, t)$, denote $\phi'_{kl}(\omega_x, \omega_y, t)$;
      Optionally, denoise by $\phi'_{kl}(\omega_x, \omega_y, t) \leftarrow \phi'_{kl}(\omega_x, \omega_y, t)\frac{A_{kl}(\omega_x, \omega_y, t)}{C(t)+\epsilon}$ as in (29);
    end
    Compute the radon transform $(\mathcal{R}\phi'_{kl})(\rho, \theta, t)$;
    Compute $PMI_{kl}(t)$ according to (33);
    if $PMI_{kl}(t) >$ *threshold* **then**
      Motion Detected at block (k, l) at time t;
      Compute direction of motion according to (34);
    end
  end
end

Algorithm 1: Phase-based motion detection algorithm using the FFT.

at the time of the video frame, block $(12, 11)$ covers a part of the vehicle in motion in the front, and block $(23, 6)$ corresponds to an area of the highway pavement where no motion occurs.

Figure 6(f) depicts, for each block (k, l), the maximum phase change over all $(\omega_x, \omega_y) \in \mathbb{D}^2$ *i.e.*,

$$\max_{(\omega_x, \omega_y) \in \mathbb{D}^2} |\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)|.$$

We observe from the figure that, for regions with low amplitude, such as the region depicting the road, when the normalization constant is absent, the derivative of the phase can be noisy. For these blocks the maximum of $|\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)|$ over all $(\omega_x, \omega_y) \in \mathbb{D}^2$ is comparable to the maximum obtained for blocks that cover the vehicles in motion.

However, equation (26) suggests that the local phase change from multiple filter pairs centered at the same spatial position (k, l) can provide a constraint to robustly estimate motion and its direction. Given the block structure employed in the computation of the local phase, it is natural to utilize phase change information from multiple sources.

Indeed, if for a particular block (k, l), $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t) = -v_x(t)\omega_x - v_y(t)\omega_y$, then it is easy to see that $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t)$ will be zero on the line $v_x(t)\omega_x + v_y(t)\omega_y = 0$ and have opposite sign on either side of this line. For example, in Figures 6(b) and 6(c), $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)$ clearly exhibits this property for blocks that cover a vehicle in motion. The PMI is a tool to evaluate this property.

Finally, the PMIs for all blocks are shown compactly in a heat map in Figure 6(e). The figure shows clearly that the blocks corresponding to the two moving vehicles have a high PMI value while the stationary background areas have a low PMI value, allowing one to easily detect motion by employing simple thresholding (see also the plane "RADON TRANSFORM AND EXTRACT STRENGTH ORIENTATION OF PLANE" in Figure 5). In addition, the orientation of motion in each block is readily observable even by inspection in Figure 6(b) by a line separating the yellow part and blue part in each block. Further results about the direction of motion are presented in Section 4.

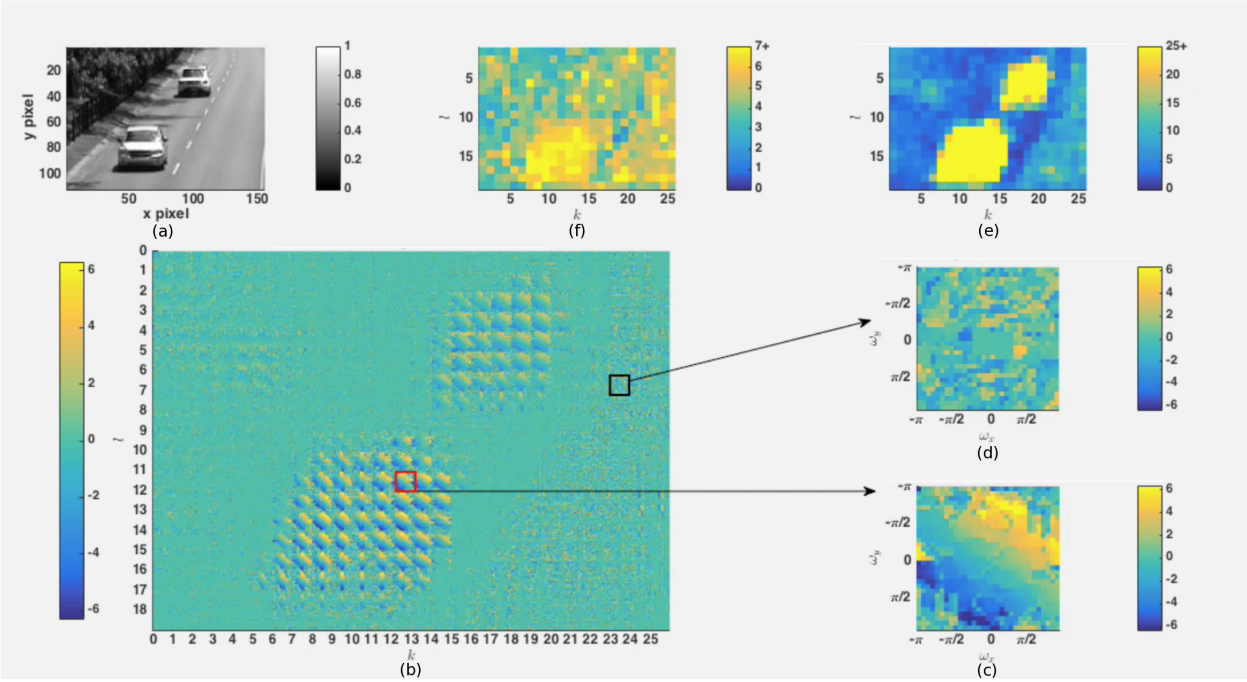


Figure 6: Evaluating the motion detection algorithm at time t_0 (see also supplementary video S1).
(a) A still from a 156×112-pixel video at time t_0.
(b) $\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)$ for all 26×19 blocks, each of size 32×32. The blocks are concatenated to create a large "phase image" with their relative neighbors kept.
(c) $\frac{d\phi_{12,11}}{dt}(\omega_x, \omega_y, t_0)$ shown in the $(\omega_x, \omega_y) \in \mathbb{D}^2$ space.
(d) $\frac{d\phi_{23,6}}{dt}(\omega_x, \omega_y, t_0)$ shown in the $(\omega_x, \omega_y) \in \mathbb{D}^2$ space.
(e) Phase Motion Indicator (PMI) for all blocks at t_0. Each pixel (k,l) is color coded for PMI_{kl}.
(f) $\max_{(\omega_x,\omega_y)\in\mathbb{D}^2} |\frac{d\phi_{kl}}{dt}(\omega_x, \omega_y, t_0)|$ for all blocks at t_0. Each pixel (k, l) is color coded for this maximum value of block (k, l).

3.4 Relationship to Biological Motion Detectors

A straightforward way to implement local motion detectors is to apply a complex-valued Gabor receptive field (5) to the video signal u, and then take the derivative of the phase

w.r.t. time or apply a highpass filter on the phase to approximate the derivative.

We present here an alternate implementation without explicitly computing the phase. This will elucidate the relation between the phase-based motion detector presented in the previous section and some elementary motion detection models used in biology, such as the Reichardt motion detector [7] and motion energy detector [9].

Assuming that the local phase $\phi_{00}(\omega_x, \omega_y, t)$ is differentiable, we have (see also Appendix A)

$$\frac{d\phi_{00}(\omega_x, \omega_y, t)}{dt} = \frac{\frac{db(\omega_x,\omega_y,t)}{dt} a(\omega_x, \omega_y, t) - \frac{da(\omega_x,\omega_y,t)}{dt} b(\omega_x, \omega_y, t)}{[a(\omega_x, \omega_y, t)]^2 + [b(\omega_x, \omega_y, t)]^2}, \tag{36}$$

where $a(\omega_x, \omega_y, t)$ and $b(\omega_x, \omega_y, t)$ are, respectively, the real and imaginary parts of $A_{00}(\omega_x, \omega_y, t)e^{j\phi_{00}(\omega_x,\omega_y,t)}$.

We notice that the denominator of (36) is the square of the local amplitude of u, and the numerator is of the form of a second order Volterra kernel. This suggests that the time derivative of the local phase can be viewed as a second order Volterra kernel that processes two *normalized* spatially filtered inputs v_1 and v_2.

We consider an elaborated Reichardt motion detector as shown in Figure 7. It is equipped with a quadrature pair of Gabor filters whose outputs are $r_1(t) = a(\omega_x, \omega_y, t)$ and $r_2(t) = b(\omega_x, \omega_y, t)$, respectively, for a particular value of (ω_x, ω_y). The pair of Gabor filters that provide these outputs are the real and imaginary parts of $w(x, y)e^{-j(\omega_x x+\omega_y y)}$. It also consists of a temporal high-pass $g_1(t)$ and temporal low-pass filter $g_2(t)$ [9]. The output of the elaborated Reichardt detector follows the diagram in Figure 7 and can be expressed as

$$(r_2 * g_1)(t)(r_1 * g_2)(t) - (r_1 * g_1)(t)(r_2 * g_2)(t), \tag{37}$$

The response can also be characterized by a second order Volterra kernel. We notice the striking similarity between (37) and the numerator of (36). In fact, the phase-based motion detector shares some properties with the Reichardt motion detector. For example, it is straightforward to see that a single phase-based motion detector is tuned to the temporal frequency of a moving sinusoidal grating.

Since the motion energy detector is formally equivalent to an elaborated Reichardt motion detector [9], the structure of the motion energy detector with divisive normalization is also similar to the phase-based motion detector.

4 Exploratory Results

In this section, we apply the phase-based motion detection algorithm on several video sequences and demonstrate its efficiency and effectiveness in detecting local motion. The motion detection algorithm is compared to two well-known biological motion detectors, namely the Reichardt motion detector and the Barlow-Levick motion detector. We then show that the detected local motion can be used in motion segmentation tasks. The effectiveness of

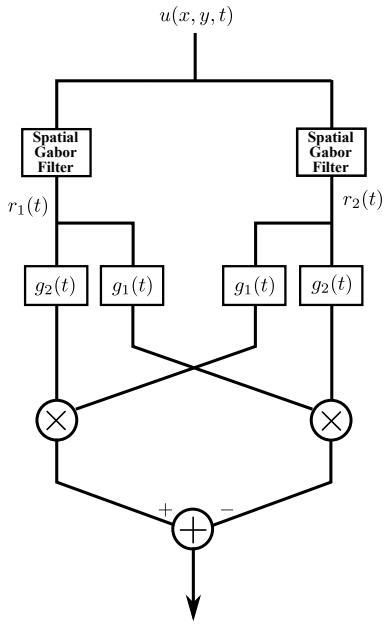


Figure 7: Diagram of an elaborated Reichardt motion detector. It consists of a quadrature pair of Gabor filters whose outputs r_1 and r_2 provide inputs to the high-pass filters $g_1(t)$ and the low-pass filters $g_2(t)$.

the segmentation is compared to segmentation using motion information obtained from a widely used optic flow based algorithm available in the literature [4].

4.1 Efficient Parallel Implementation

The algorithm was implemented in PyCUDA [23] and tested on an NVIDIA GeForce GTX TITAN GPU. All computation uses single precision floating points. The processing speeds of the algorithm for several screen sizes are listed in Table 1. Clearly, the proposed phase-based motion detection algorithm has real-time capability to process video even with full High Definition screen size.

Table 1: Processing Speeds of the Proposed Algorithm, the Reichardt motion detector and the Barlow-Levick Motion detector

Screen Size	Processing Capability (Frames Per Second)		
	Proposed	Reichardt	Barlow-Levick
320×240	420	790	800
720×576	135	685	690
1280×720	60	274	293
1920×1080	27	191	194

For comparison, we implemented the Reichardt motion detector and the Barlow-Levick motion detector. Their respective diagrams are shown in Figure 8. Note that we moved the

high-pass filter $h_1(t)$ to the front of the low-pass filter $h_2(t)$. This configuration provides a superior performance to the one in Figure 7. For both the Reichardt and the Barlow-Levick detectors, the videos are first blurred by a Gaussian filter (with the same variance as the Gaussian window in (27) used for the phase-based motion detector) and subsampled at the center of each overlapping block in the phase-based motion detector. The subsampled video then provide inputs to two 2D arrays of the circuits shown in Figure 8, one for the horizontal direction and one for the vertical direction. The outputs of the horizontal and vertical motion circuits form an array of motion vectors that indicate the strength and direction of motion. The three tested motion detectors have the same number of outputs as a result. The processing speeds of the Reichardt motion detector and the Barlow-Levick motion detector, both implemented in PyCUDA and tested on the same GPU, are shown in Table 1.

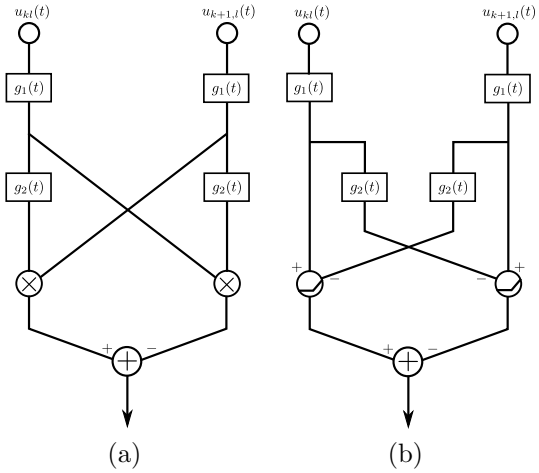


Figure 8: (a) A Reichardt motion detector detecting motion between the two points u_{kl} and $u_{k+1,l}$ in the blurred and down-sampled video. (b) A Barlow-Levick motion detector detecting motion between the two points u_{kl} and $u_{k+1,l}$ in the blurred and down-sampled video. $g_1(t)$ denotes a high-pass filter and $g_2(t)$ denotes a low-pass filter.

Note that the Reichardt motion detector and the Barlow-Levick motion detector are highly efficient due to the simplicity of their algorithms. The phase-based motion detection algorithm, however, is a much more sophisticated algorithm, and yet it can be implemented in real-time using parallel computing devices.

The fast GPU implementation is based on the FFT and Matrix-Matrix multiplication. It is expected that those operations can be efficiently implemented in hardware, *e.g.,* FPGA.

4.2 Examples of Phased-Based Motion Detection

We applied our motion detection algorithm on video sequences of the ChangeDetection2014 dataset [22] that did not exhibit camera egomotion. For these video sequences, the standard deviation of the Gaussian window functions was set to 4 pixels and the block size was chosen to be 32×32 pixels. Threshold and normalization parameters were kept the same with the

exception of the "thermal video" (in order to deal with larger background noise levels, see below). We also tested the same video sequences using the Reichardt motion detector and the Barlow-Levick motion detector. For the Reichardt detector, the high-pass filters were chosen as first order filters with a time constant of 200 milliseconds, and the low-pass filters were chosen as first order filters with a time constant of 300 milliseconds (assuming that the frame rate is 50 frames per second). Threshold was set to 2. For the Barlow-Levick motion detector, the time constant of the first high-pass filters were set to 250 milliseconds. The low-pass filters were the same as in the Reichardt detector. Threshold was set to 2.

The first video was taken from a highway surveillance camera ("highway video") under good illumination conditions and high contrast. The video had moderate noise, particularly on the road surface. The detected motion is shown in the top left panel of Figure 9 (see supplementary Video S2 for full video). The phase-based motion detection algorithm captured both the moving cars and the tree leaves moving (due to the wind). In the time interval between the 9th and 10th second, the camera was slightly moved left- and right-wards within 5 frames, again possibly due to the wind. Movement due to this shift was captured by the motion detection algorithm and the algorithm was fast enough to correctly determine the direction of this movement. In this video, we already noted that this algorithm suffers from the aperture problem. For examples, in front of the van where a long, horizontal edge is present, the detected motion is mostly pointing downwards. In addition to moving downwards, the edge is also moving to the left, however. This is expected since the algorithm only detects motion locally and does not take into account the overall shape of any object.

For comparison, the motion detection results for the Reichardt motion detector and the Barlow-Levick motion detector are shown in the top middle and top right panel of Figure 9 (see supplementary Video S2 for full video). The Reichardt detector performed relatively well when vehicles are moving faster, but the direction it predicts for vehicles moving slower, *e.g.*, on the back of the image is not accurate. In addition, motion is still detected in some parts of the screen where the vehicles have just passed by. The detection result for the Barlow-Levick motion detector was poorer. In particular, the response to OFF edge movement is always the opposite to the actual movement direction.

We then squeezed the range of the screen intensity from $[0, 1]$ to $[0.2, 0.4]$, resulting in a video with a lower mean luminance and lower contrast. The motion detection results on the low-contrast video are shown in the bottom 3 panels of Figure 9 (see supplementary Video S2 for full video). For reference, the motion detection results for the original video are shown in red arrows. Motion detected in the low-contrast video is shown in blue arrows if no motion is detected for the block in the original video. If motion is detected in both the original and the low-contrast video, the arrows are shown in magenta. Figure 9 clearly shows that while the motion detection performance is degraded for all three detectors, the phase-based motion detector performed still quite well in the low-contrast video, detecting most of the moving vehicles. The other two detectors missed many of the blocks where motion was detected in the original movie.

To quantify how well each detector works under different contrast conditions, we computed the ratio of unthresholded output values of each motion detector between lower contrast and full contrast video. For example, in the case of phase-based motion detector, we computed,

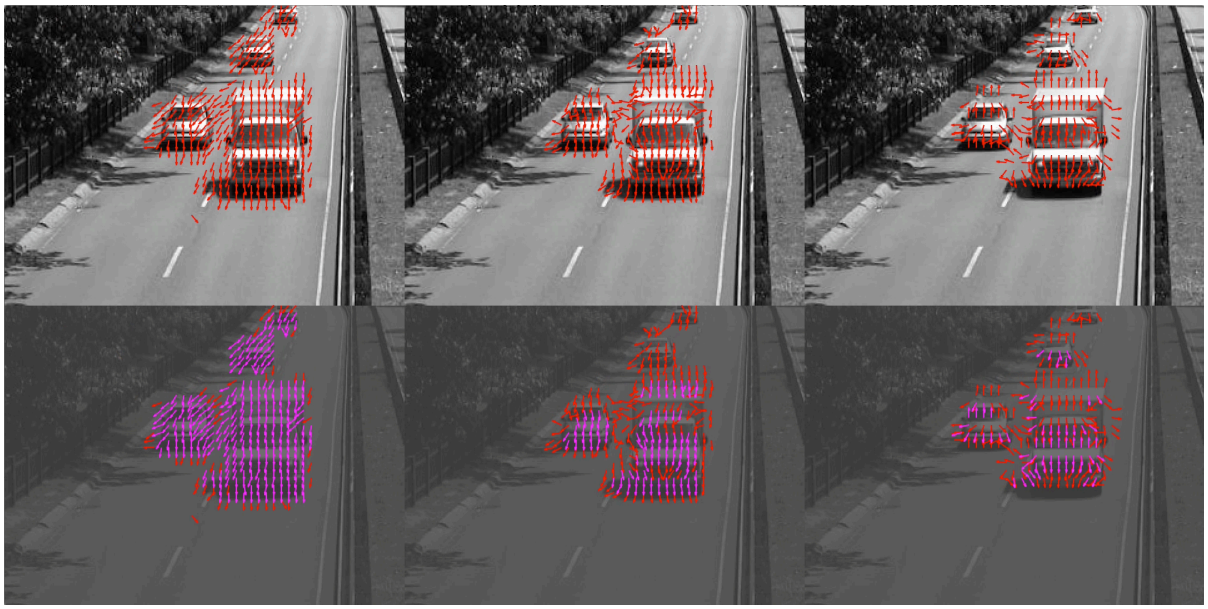


Figure 9: (Top) Motion detection of the "highway video" using the phase-based algorithm (left), the Reichardt motion detector (middle), and the Barlow-Levick motion detector (right). Red arrows indicate detected motion and its direction. (Bottom) The contrast of the video was artificially reduced by 5 folds and the mean was reduced to 3/5 of the original. The red arrows are duplicated from the motion detection result on the original video as in Top. Blue arrows are the result of motion detection on the video with reduced contrast. If motion is detected both in the original video and in the video with reduced contrast, then the arrow is shown in magenta (as a mix of blue and red).

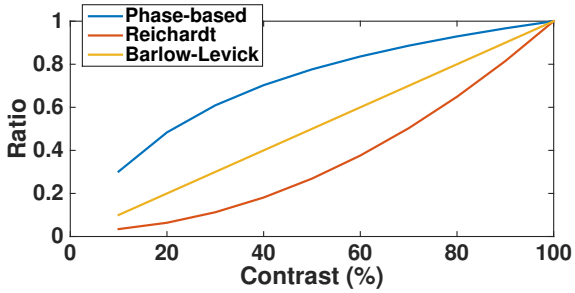


Figure 10: The ratio for the phase-based motion detector is defined as the following: we set the motion detected in the 100% contrast case as the baseline; we compute the ratio between the PMI index for the lower contrast video and that for the full contrast video for each block; the ratios are averaged across all the blocks where motion is detected in the full contrast video; the average is given as the ratio for the phase-based motion detector. The ratio for the Reichardt motion detector and the Barlow-Levick motion detector are similarly defined.

for each block, the ratio between the PMI index for the lower contrast video and that for the full contrast video. The ratios are then averaged across all blocks where motion is detected in the full contrast video. This average for different contrasts are shown in Figure 10 by the blue curve. In the ideal case when the normalization constant ϵ is 0, the phase detector should produce invariant responses to videos with different contrast. The curve shown here is mainly due to a non-zero ϵ. The ratios for the Reichardt motion detector and the Barlow-Levick motion detector are computed similarly and are shown in red and yellow, respectively. It is clear that the phase-based motion detector has a superior performance across a range of contrast values. At 20% contrast, the phase-based detector still has 50% of the PMI index value for full contrast video. As expected, the response of Barlow-Levick motion detector is linear with respect to contrast, and the Reichardt motion detector has a quadratic relation to contrast. For a fixed threshold value a larger ratio equates to a more consistent performance at lower contrast.

The second video was captured by a surveillance camera in a train station ("train station video"). The video was under moderate room light with a low noise level. The front side of the video had high contrast; illumination on the back side was quite low. The detected motion is shown in the video of Figure 11 (see supplementary Video S3 for full video).. Movements of people were successfully captured by the motion detection algorithm.

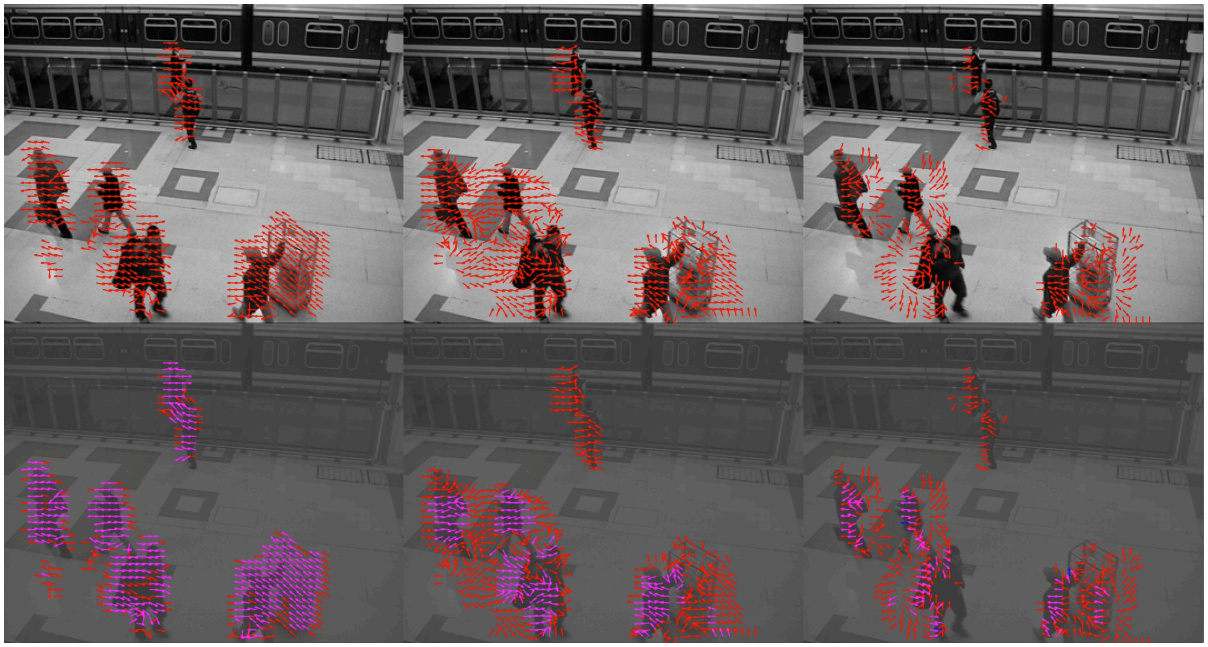


Figure 11: Motion detection of the "train station video" using the phase-based algorithm (left), the Reichardt motion detector (middle) and the Barlow-Levick motion detector (right) (see Figure 9 for description of each panel).

The third video was a "thermal video" with large amount of background noise (thermal video). The threshold for detecting motion was raised by 60% in order to mitigate the increased level of noise. The detected motion is shown in the video of Figure 12 (see supplementary Video S4 for full video).

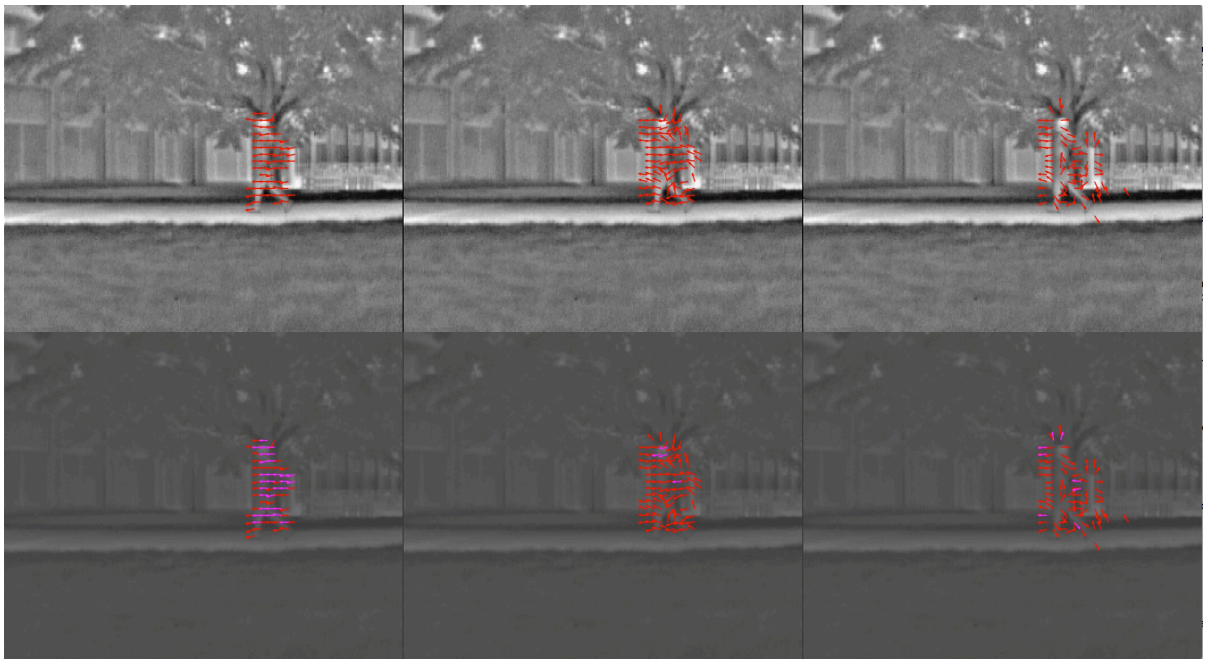


Figure 12: Motion detection of the "thermal video" using the phase-based algorithm(left), the Reichardt motion detector (middle) and the Barlow-Levick motion detector (right) (see Figure 9 for description of each panel).

The last example we show here was taken from a highway surveillance camera at night ("winterstreet video"). The overall illumination on the lower-left side was low whereas illumination was moderate on the upper-right side where the road was covered by snow. The detected motions are shown in the video in Figure 13 (see supplementary Video S5 for full video). We note that, overall, car movements were successfully detected. Car movements on the lower-left side, however, suffered from low illumination and some parts of the car were not detected well due to the trade-off employed for noise suppression.

With a higher threshold, the phase-base motion detection algorithm is able to detect motion under noisy conditions. We added to the original "highway video" and "train station video" Gaussian white noise with standard deviation 5% of the maximum luminance range. The results are shown, respectively, in Figure 14 and in Figure 15 (see supplementary Videos S6 and S7 for full videos).

4.3 Examples of Motion Segmentation

We asked whether the detected motion signals in the video sequences can be useful for segmenting moving objects from the background. We applied a larger threshold to only signal motion for salient objects. The 32×32 blocks, however, introduce large boundaries around the moving objects. To reduce the boundary and to segment the moving object more closely to the actual object boundary , we applied the motion detection algorithm around the detected boundary with 16×16 blocks. If 16×16 blocks did not indicate motion, then

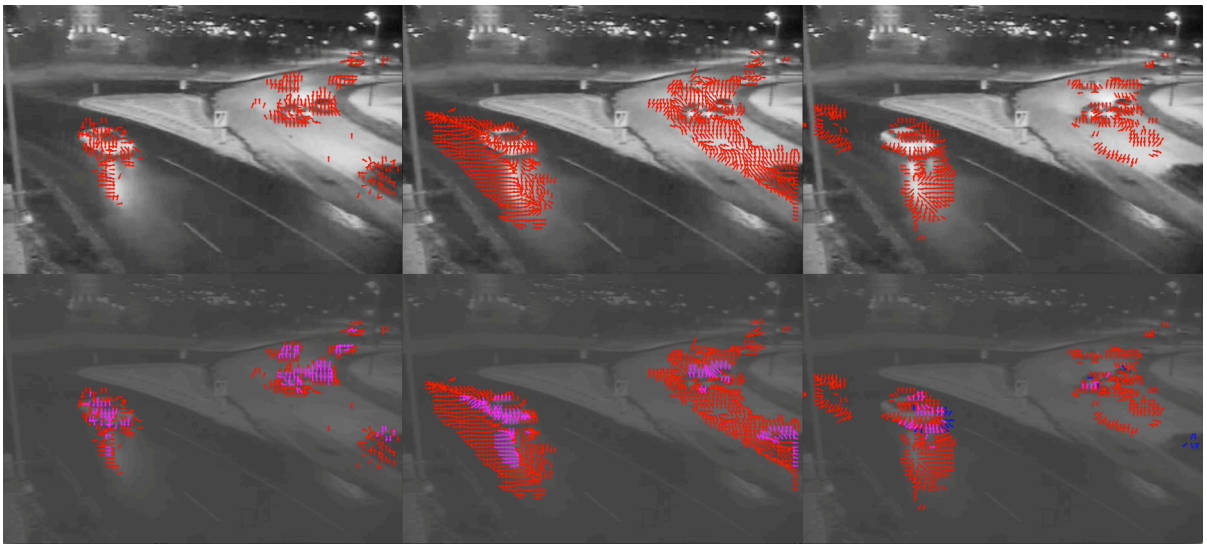


Figure 13: Motion detection of the "winterstreet video" using the phase-based algorithm(left), the Reichardt motion detector (middle) and the Barlow-Levick motion detector (right) (see Figure 9 for description of each panel).

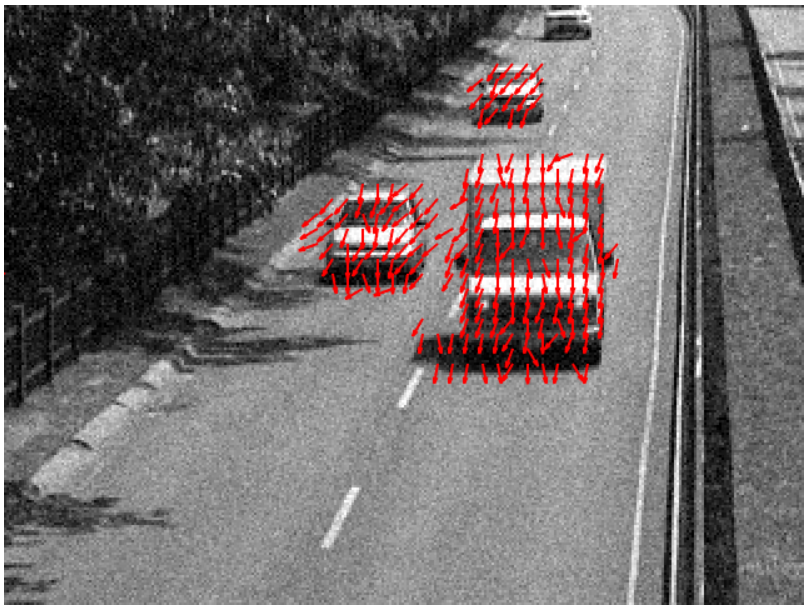


Figure 14: Phase-based motion detection applied on the "highway video" with added Gaussian white noise.

Figure 15: Phase-based motion detection applied on the "train station video" with added Gaussian white noise.

the corresponding area was removed from the segmented object area.

For comparison, we performed motion segmentation based on local motion detection based on an optic flow algorithm [4]. The segmentation in this case was implemented by comparing the length of the optic flow vectors with an appropriate threshold.

We employed a simple thresholding for both phase-based motion detection algorithms and optic flow based motion detection. More sophisticated algorithms may produce better segmentation results. Thus, the segmentation was purely based on motion cues and no post-processing at pixel level was performed. The state-of-the-art results for the ChangeDetection2014 dataset utilize multiple cues such as motion, color and background extraction to segment objects and thereby achieve better results. We are exploring here, however, only the case where motion is the only cue for segmentation. Therefore, the ground truth information from the dataset was not applicable to our test. We shall, therefore, only show the effectiveness of motion segmentation visually.

We first applied motion based segmentation on 2-second segment of the "highway video". The result using the local phase-based motion detection algorithm is shown in the video of Figure 16a (see supplementary Video S8a for full video)., and that using optic flow based motion detection algorithm is shown in the video of Figure 16b (see supplementary Video S8b for full video).. Both videos are played back at 1/4 of speed. With a higher threshold, the movement of the leaves was no longer picked up by the phase-based motion detector. Therefore, only the cars were identified as moving objects and they are indicated in red.

Although the moving objects were not perfectly segmented on their boundary, they were mostly captured. For the optic flow based segmentation, since the regions of interest are set by thresholding the length of the velocity vector, objects moving at lower speed, *e.g.*, the cars on the top, were not always picked up by the segmentation algorithm.

We then applied the motion segmentation to a 2-second segment of the "train station video", the "thermal video" and the "winterstreet video". The results are shown, respectively, in the videos of Figure 17, Figure 18 and Figure 19 (see supplementary Videos S9a and S9b, S10a and S10b, S11a and S11b respectively for full videos).

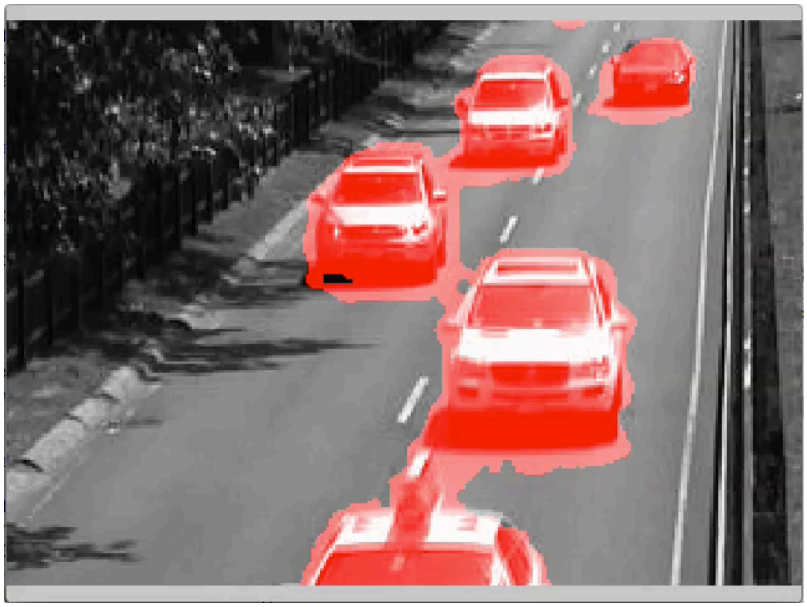
These results show that for the purpose of detecting local motion and its use as a motion segmentation cue, the local phase-based motion detector works as good, if not better than a simple thresholding segmentation using an optic flow based algorithm.

5 Discussion

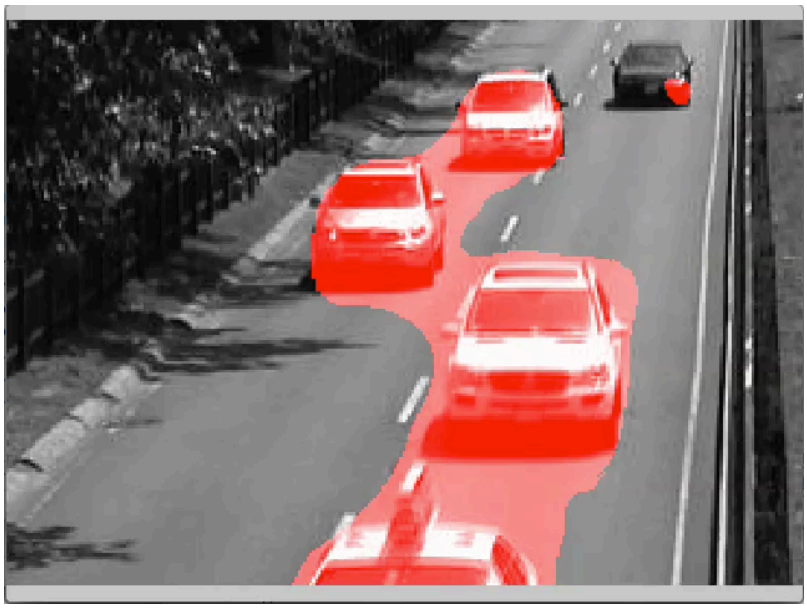
Previous research demonstrated that *global* phase information alone can be used to faithfully represent visual scenes. Here we provided a reconstruction algorithm of visual scenes by only using *local* phase information. More importantly, local phase information can be effectively used to detect local motion. Through a simple temporal derivative of the phase, we obtained a second order Volterra kernel that is applied on two normalized inputs. The structure of the second order Volterra kernel in the phase-based motion detector is akin to models employed to detect motion in biological vision, e.g., the Reichardt detector [7] and the motion energy detector [9].

We then proposed an efficient, FFT-based algorithm employing the change in local phase for detecting motion. In order to exploit the special structure of the change in phase in the frequency domain that is due to rigid motion, the phase-based motion detection algorithm also incorporates the Radon transform, a transform closely related to the Fourier transform. Based on the Radon transform, a motion indicator was proposed to robustly detect whether the phase change is caused by motion. Therefore, the algorithm can be efficiently implemented whenever the FFT is available/supported. We showed examples of applying the phase-based motion detection algorithm on several video sequences. We also showed that the locally detected motion can be used for segmenting moving objects in video scenes. We compared the segmentation of moving objects using our local phase-based algorithm to segmentation achieved using a widely used optic flow based algorithm. Our results suggest that spatial phase information may provide an efficient alternative to perform many visual tasks in silico as well as in modeling in vivo biological vision systems. This is consistent with other recent findings [15].

Note that phase information has been used for solving various visual tasks in the past. In fact, phase has been successfully employed in optic flow algorithms [24] and image registration for translation [25], both applied to motion related tasks. The phase-based optic flow algorithm applies the optic flow equation on the local phase of images rather than the intensity itself to achieve better resolution and robustness in estimating motion velocity. The phase correlation

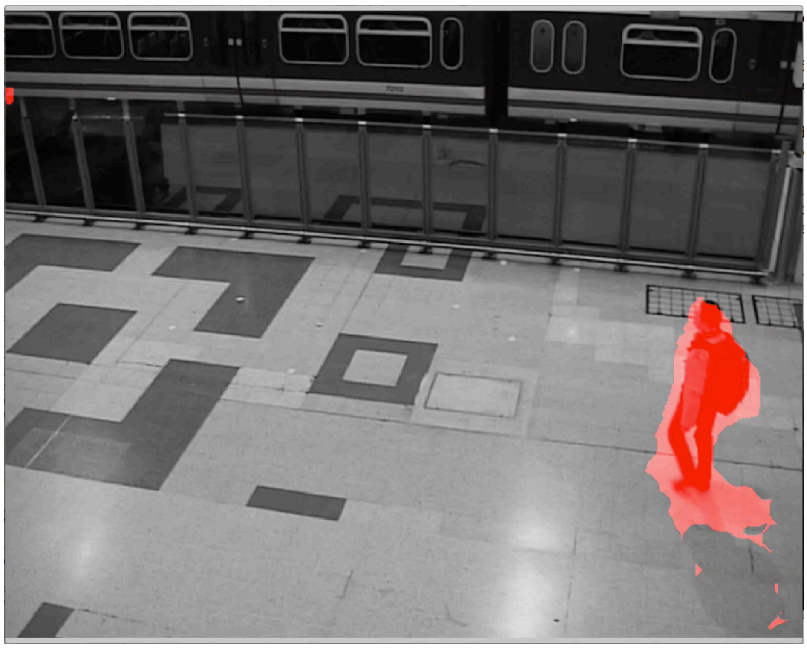


(a)



(b)

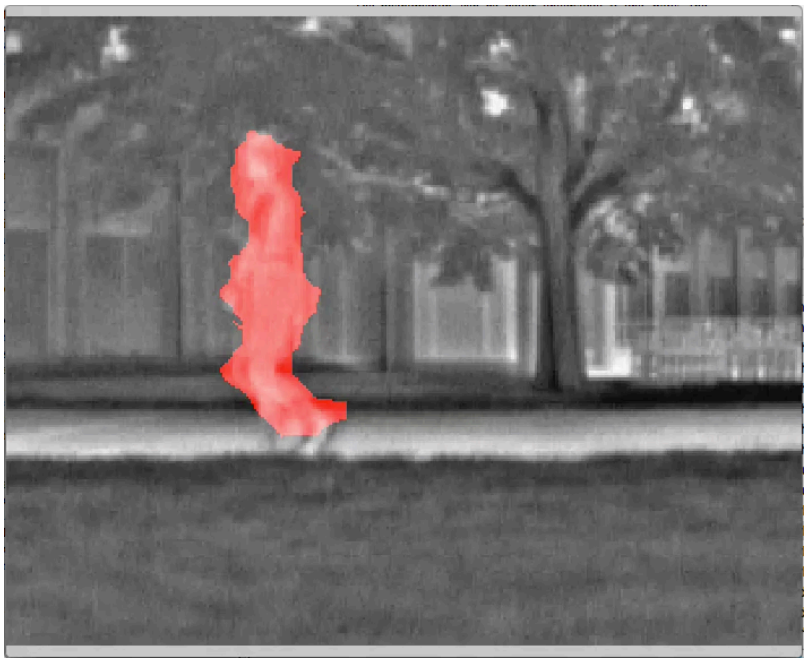
Figure 16: Segmentation of moving cars based on detected motion in the "highway video". (a) Motion detected by the phase-based motion detection algorithm. (b) Motion detected using optic flow algorithm [4].



(a)

(b)

Figure 17: Segmentation of moving people in the "train station video" based on detected motion. (a) Motion detected by the phase-based motion detection algorithm. (b) Motion detected using optic flow algorithm [4].



(a)

(b)

Figure 18: Segmentation of moving people in the "thermal video" based on detected motion. (a) Motion detected by the phase-based motion detection algorithm. (b) Motion detected using optic flow algorithm [4].

(a)

(b)

Figure 19: Segmentation of moving cars in the "winter street video" based on detected motion. (a) Motion detected by the phase-based motion detection algorithm. (b) Motion detected using optic flow algorithm [4].

method computes the normalized cross-power spectrum of two images to extract the phase difference. It provides better accuracy and robustness as compared to the classical cross correlation approach applied to two consecutive images in a sequence. However, it has limitations when dealing with images with a repetitive structure.

Our method differs from the above two cases in the following ways. First, it employs a simple temporal derivative/highpass filtering on the phase to extract local phase changes. The structure of the *phase change* is exploited for better detection of motion. On the contrary, the phase correlation method considers the structure of the *phase itself*. This also allowed us to make the motion detection a more local, continuous process rather than purely operating globally on discrete frames. Second, it explores the structure of the change of phase due to motion in the frequency domain rather than in the spatial domain, in which the key constraints of optic flow equations are based. Third, instead of focussing on estimating the exact velocity or shift, our method is centered on the detection of motion with a coarse local estimate of direction. This is the case in the first steps of biological motion detection in the retina or optic lobe of insects; we discussed the resemblance of the method presented here to those of biological models of motion detection.

The proposed motion detection algorithm, however, shares several advantages with the other phase-based methods. For example, it is sensitive to motion that only induces a subpixel shift between frames and for very small differences in intensity. In addition, when compared to the amplitude, the local phase is robust under different contrast and illumination conditions. Consequently, the algorithm presented here can operate in a wide range of contrast/illumination conditions.

Furthermore, once the local phase is extracted from each block, motion detection becomes a localized, temporal operation on the local phase of each block. This forms the basis for the highly parallel structure in the phase-based motion detection algorithm. By contrast, traditional optic flow techniques often rely on explicit comparisons across spatial locations, that increase, for example, memory complexity since all states must be made available to their neighbors.

We also notice that for each block, a large number of measurements of phase changes are obtained. From Figure 6, we see that this number is much higher than that of the original pixel space. In other words, in order to detect motion in a robust way, the local phase-based motion detection algorithm undergoes an expansion of measurements before settling down onto a single motion indicator value. This number of measurements, however, does not incur additional computational demand thanks to the highly efficient FFT algorithm. Biological visual systems often have a similar structure. For example, in the vertebrate retina, computations are carried out by an extraordinarily large number of neurons until measurements of the visual scene are projected by a fraction of the neurons onto the cortex [26]. Similar expansion takes places in the primary visual cortex as well.

We also highlight the ease of implementation of the intrinsically parallel algorithm proposed here. The algorithm introduced in Section 3.3 is based on the FFT algorithm and does not require solving an optimization problem. It can be efficiently implemented in hardware, *e.g.,* FPGAs, or in software, *e.g.*, using GPU accelerators. We note that extending the FFT to

higher dimensions is straightforward and the implementations of higher dimensional FFTs are also highly efficient. Clearly, the methodology can be applied to motion detection of data in 3D or higher dimensional space, where the Radon transform operates over planes or hyperplanes.

Finally, we argue that the change of phase, although highly nonlinear, can be obtained through a normalization (gain control) followed by a second order Volterra kernel. This separation of a higher order nonlinearity into gain control block and a lower order nonlinear filter can be used for modeling motion detection circuits in biological systems.

6 Acknowledgements

The research reported here was supported by AFOSR under grant #FA9550-12-1-0232.

7 Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

References

[1] David Marr. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. The MIT Press, 2010.

[2] Joshua R Sanes and S Lawrence Zipursky. Design principles of insect and vertebrate visual systems. Neuron, 66(1):15–36, 2010.

[3] Alexander Borst. In search of the holy grail of fly motion vision. European Journal of Neuroscience, 40(9):1–9, 2014.

[4] 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, June 2010.

[5] Florian Raudies. Optic flow. Scholarpedia, 8(7):30724, 2013.

[6] Garrick Orchard and Ralph Etienne-Cummings. Bioinspired visual motion estimation. Proceedings of the IEEE, 102(10):1520–1536, 2014.

[7] Bernhard Hassenstein and Werner Reichardt. Systemtheoretische analyse der zeit-, reihenfolgen- und vorzeichenauswertung bei der bewegungsperzeption des rüsselkäfers chlorophanus,. Z. Naturforsch. B, 11(9):513–524, 1956.

[8] Alexander Borst, Juergen Haag, and Dierk F. Reiff. Fly motion vision. Annual Review Neuroscience, 33:49–70, 2010.

[9] Edward H. Adelson and James R. Bergen. Spatiotemporal energy models for the perception of motion. Journal of Optical Society of America. A, Optics and Image Science, 2(2):284–299, 1985.

[10] H. B. Barlow and W. R. Levick. The mechanism of directionally selective units in rabbit's retina. The Journal of Physiology, 178(3):477, 1965.

[11] Alan V. Oppenheim and Jae S. Lim. The importance of phase in signals. Proceedings of the IEEE, 69(5):529–541, 1981.

[12] T. Gerkmann, M. Krawczyk-Becker, and J. Le Roux. Phase processing for single-channel speech enhancement: History and recent advances. Signal Processing Magazine, IEEE, 32(2):55–66, March 2015.

[13] Mohammed Kadiri, Mohamed Djebbouri, and Philippe Carré. Magnitude-phase of the dual-tree quaternionic wavelet transform for multispectral satellite image denoising. EURASIP Journal on Image and Video Processing, 2014(1):1–16, 2014.

[14] Peter Kovesi. Image features from phase congruency. Videre: Journal of Computer Vision Research, 1(3), 1999.

[15] Evgeny Gladilin and Roland Eils. On the role of spatial phase and phase correlation in vision, illusion, and cognition. Frontiers in Computational Neuroscience, 9(45):1–14, 2015.

[16] Peter Dayan and L.F. Abbott. Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT Press, Cambridge, MA, 2001.

[17] Tai Sing Lee. Image representation using 2d gabor wavelets. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(10):959–971, 1996.

[18] Monson H. Hayes, Jae S. Lim, and Alan V. Oppenheim. Signal reconstruction from phase or magnitude. IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(6):672–680, 1980.

[19] Emmanuel J. Candès, Thomas Strohmer, and Vladislav Voroninski. PhaseLift: Exact and stable signal recovery from magnitude measurements via convex programming. Communications in Pure and Applied Mathematics, 66(8):1241–1274, 2011.

[20] Aurel A Lazar, Eftychios A Pnevmatikakis, and Yiyin Zhou. The power of connectivity: Identity preserving transformations on visual streams in the spike domain. Neural Networks, 44:22–35, 2013.

[21] Aurel A Lazar and Yiyin Zhou. Volterra dendritic stimulus processors and biophysical spike generators with intrinsic noise sources. Frontiers in computational neuroscience, 8, 2014.

[22] Yi Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on, pages 393–400, June 2014.

[23] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, B. Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: A Scripting-Based Approach to GPU Run-Time Code Generation. Parallel Computing, 38(3):157–174, 2012.

[24] David J. Fleet and Allan D. Jepson. Computation of component image velocity from local phase information. International Journal of Computer Vision, 5(1):77–104, 1990.

[25] E. De Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5):700–703, 1987.

[26] Chang-Jin Jeon, Enrica Strettoi, and Richard H. Masland. The major cell populations of the mouse retina. Journal of Neuroscience, 18(21):8936–8946, 1998.

A The Derivative of the Local Phase

Let

$$a(\omega_x, \omega_y, t) = \int_{\mathbb{R}^2} u(x, y, t) w(x, y) \cos(\omega_x x + \omega_y y) dx dy \tag{38}$$

and

$$b(\omega_x, \omega_y, t) = -\int_{\mathbb{R}^2} u(x, y, t) w(x, y) \sin(\omega_x x + \omega_y y) dx dy, \tag{39}$$

and therefore,

$$A_{00}(\omega_x, \omega_y, t) e^{j\phi_{00}(\omega_x, \omega_y, t)} = a(\omega_x, \omega_y, t) + jb(\omega_x, \omega_y, t). \tag{40}$$

The local phase amounts to $\phi_{00}(\omega_x, \omega_y, t) = \arctan2(b(\omega_x, \omega_y, t), a(\omega_x, \omega_y, t))$ and therefore

$$\frac{d\phi_{00}}{dt} = \frac{a\frac{db}{dt} - b\frac{da}{dt}}{[a]^2 + [b]^2}. \tag{41}$$

Now,

$$\frac{da}{dt}(\omega_x, \omega_y, t) = \int_{\mathbb{R}^2} \frac{\partial u}{\partial t}(x, y, t) w(x, y) \cos(\omega_x x + \omega_y y) dx dy. \tag{42}$$

For a purely translational signal with instantaneous spatial shift given by $(s_x(t), s_y(t))$ within the support of the window we have,

$$\begin{aligned}
\frac{da}{dt}(\omega_x, \omega_y, t) &= -\int_{\mathbb{R}^2} \frac{ds_x}{dt}(t) \frac{\partial u}{\partial x}(x, y, t) w(x, y) \cos(\omega_x x + \omega_y y) dx dy \\
&\quad - \int_{\mathbb{R}^2} \frac{ds_y}{dt}(t) \frac{\partial u}{\partial y}(x, y, t) w(x, y) \cos(\omega_x x + \omega_y y) dx dy \\
&= \frac{ds_x}{dt} \int_{\mathbb{R}^2} u(x, y, t) \frac{\partial w}{\partial x}(x, y) \cos(\omega_x x + \omega_y y) dx dy \\
&\quad - \omega_x \frac{ds_x}{dt} \int_{\mathbb{R}^2} u(x, y, t) w(x, y) \sin(\omega_x x + \omega_y y) dx dy \\
&\quad + \frac{ds_y}{dt} \int_{\mathbb{R}^2} u(x, y, t) \frac{\partial w}{\partial y}(x, y) \cos(\omega_x x + \omega_y y) dx dy \\
&\quad - \omega_y \frac{ds_y}{dt} \int_{\mathbb{R}^2} u(x, y, t) w(x, y) \sin(\omega_x x + \omega_y y) dx dy \\
&= \frac{ds_x}{dt} \underbrace{\int_{\mathbb{R}^2} u(x, y, t) \frac{\partial w}{\partial x}(x, y) \cos(\omega_x x + \omega_y y) dx dy}_{r_3 = r_3(\omega_x, \omega_y, t)} + \omega_x \frac{ds_x}{dt} b \\
&\quad + \frac{ds_y}{dt} \underbrace{\int_{\mathbb{R}^2} u(x, y, t) \frac{\partial w}{\partial y}(x, y) \cos(\omega_x x + \omega_y y) dx dy}_{r_4 = r_4(\omega_x, \omega_y, t)} + \omega_y \frac{ds_y}{dt} b.
\end{aligned} \tag{43}$$

Similarly,

$$\begin{aligned}\frac{db}{dt}(\omega_x,\omega_y,t) = &-\frac{ds_x}{dt}\underbrace{\int_{\mathbb{R}^2} u(x,y,t)\frac{\partial w}{\partial x}(x,y)\sin(\omega_x x+\omega_y y)dxdy}_{r_5=r_5(\omega_x,\omega_y,t)} - \omega_x\frac{ds_x}{dt}a \\ &- \frac{ds_y}{dt}\underbrace{\int_{\mathbb{R}^2} u(x,y,t)\frac{\partial w}{\partial y}(x,y)\sin(\omega_x x+\omega_y y)dxdy}_{r_6=r_6(\omega_x,\omega_y,t)} - \omega_y\frac{ds_y}{dt}a.\end{aligned} \tag{44}$$

Plugging in the value of $\frac{da}{dt}$ and $\frac{db}{dt}$, the derivative of the local phase amounts to

$$\begin{aligned}\frac{d\phi_{00}}{dt}(\omega_x,\omega_y,t) &= \frac{\begin{array}{r}-\frac{ds_x}{dt}\left[r_3 b + r_5 a + \omega_x[a]^2 + \omega_x[b]^2\right] \\ -\frac{ds_y}{dt}\left[r_4 b + r_6 a + \omega_y[a]^2 + \omega_y[b]^2\right]\end{array}}{[a]^2+[b]^2} \\ &= -\frac{ds_x}{dt}\omega_x - \frac{ds_y}{dt}\omega_y \underbrace{-\frac{ds_x}{dt}\cdot\frac{r_3 b + r_5 a}{[a]^2+[b]^2} - \frac{ds_y}{dt}\cdot\frac{r_4 b + r_6 a}{[a]^2+[b]^2}}_{\mathfrak{v}_{00}(\omega_x,\omega_y,t)}.\end{aligned} \tag{45}$$