<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Block-wise motion detection using compressive imaging system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Ke, J; Ashok, A; Neifeld, MA</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Optics Communications, 2011, v. 284 n. 5, p. 1170-1180</td>
</tr>
<tr>
<td><strong>Issued Date</strong></td>
<td>2011</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10722/139225">http://hdl.handle.net/10722/139225</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>NOTICE: this is the author’s version of a work that was accepted for publication in Optics Communications. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Optics Communications, 2011, v. 284 n. 5, p. 1170-1180. DOI: 10.1016/j.optcom.2010.11.028; This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.</td>
</tr>
</tbody>
</table>
Block-wise Motion Detection Using Compressive Imaging System

Jun Ke\textsuperscript{a,\,*}, Amit Ashok\textsuperscript{a}, Mark A. Neifeld\textsuperscript{a,b}

\textsuperscript{a}\textit{Department of Electrical Computer Engineering, University of Arizona, Tucson, AZ 85721-0104, USA}

\textsuperscript{b}\textit{College of Optical Sciences, University of Arizona, Tucson, AZ 85721-0104, USA}

Abstract

A block-wise motion detection strategy based on compressive imaging, also referred to as feature-specific imaging (FSI), is described in this work. A mixture of Gaussian distributions is used to model the background in a scene. Motion is detected in individual object blocks using feature measurements. Gabor, Hadamard binary and random binary features are studied. Performance of motion detection methods using pixel-wise measurements is analyzed and serves as a baseline for comparison with motion detection techniques based on compressive imaging. ROC (Receiver Operation Characteristic) curves and AUC (Area Under Curve) metrics are used to quantify the algorithm performance. Because a FSI system yields a larger measurement SNR (Signal-to-Noise Ratio) than a traditional system, motion detection methods based on the FSI system have better performance. We show that motion detection algorithms using Hadamard and random binary features in a FSI system yields AUC values of 0.978 and 0.969 respectively. The pixel-based methods are only able to achieve a lower AUC value of 0.627.

\textit{Key words:} Compressive imaging, Feature specific imaging, Motion Detection,

\textit{Preprint submitted to Optics Communications} 25 August 2010
1 Introduction

Motion detection is an integral part of the object tracking problem. Object tracking has received widespread interest from several research communities over the years. It has broad applications spanning civilian, military, and scientific research domains. Examples include traffic control, security surveillance, bio-imaging, battlefield monitoring, and sub-sea video processing [1, 2, 3, 4, 5, 6]. In radar signal processing, Ground Moving Target Indicator (GMTI) radar has proved to be a very successful example of object tracking [7, 8]. In computer vision, an object tracking problem is viewed as a two-part problem: motion prediction and object matching problem [3]. Kalman filter [9] and particle filter [10] are among the most popular algorithms devised for the motion prediction problem [11, 2, 5]. To solve the object matching problem many different strategies have been developed such as, extracting features, calculating optical flows [3, 5, 11, 2], using rigid shapes [3, 5, 11], and matching deformable contours [12, 13]. Besides the motion prediction and object matching, another critical step in object tracking problem involves motion detection, which typically entails modeling the background or clutter, then subtracting it from a frame in a video sequence to find the moving regions [14, 15, 5, 11, 16]. Note that solving the motion detection problem can be
thought of as equivalent to solving the background subtraction problem. Proposed by Wren et al. [17], one of the earliest approaches that have been developed uses a single Gaussian distribution to model a static background at each pixel position. Subsequently, Grimson et al. developed a more sophisticated model that uses a mixture of Gaussian distribution to describe the individual pixel values in outdoor scenes [18]. In this model for motion detection, a pixel in the current frame is compared with the several Gaussian distributions to find a match. If the pixel measurement can not be matched to any distribution, it is classified as a foreground pixel and thus motion is detected. This method is fast and can be easily adjusted for clutter or illumination changes.

To further extend this Gaussian mixture model, Elgammal et al. [19] used a non-parametric kernel density estimation to describe a pixel intensity. Other methods for background subtraction include incorporating region-based scene information instead of only using single pixel information and representing the intensity variation of a pixel in a video sequence as discrete states in Hidden Markov Models (HMM) [11, 16].

Note that nearly all the methods reviewed here are based on pixel level image measurements. It is common for a digital camera today to yield a multi-megapixel image. Therefore, the pixel-wise model approach can quickly become computationally expensive. In most cases, a moving object extends over multiple pixels, therefore the motion exits at a group of adjoining pixel positions. To detect motion in a region we can not assume pixel-wise motion detection strategy is the best approach. In this work, we discuss a strategy to detect motion in a region directly. A region is defined as an object block. After discussing motion detection in object block space and its corresponding feature space, we introduce a system which makes measurements directly in
the feature space. This measurement scheme is *compressive* as the number of measurements, each measurement corresponds to one feature, is typically much smaller than the dimensionality of the object block. Such a system is referred to as a feature-specific imaging (FSI) system or compressive imaging system in some research community. The paper is organized as follows: in section 2, motion detection methods with pixel-wise measurements are discussed. We first introduce the Gaussian mixture model for individual pixels. Then the model is extended to a block-wise mixture model for an object block and a corresponding mixture model in the feature space of each block. In section 3, simulation results for the detection methods described in section 2 are presented. In section 4, we define the FSI system using the parallel optical architecture followed by a description of motion detection algorithms designed for FSI measurements. In section 5, we present the simulation results for the motion detection methods using FSI system. Finally, in section 6 we draw conclusions based on the results of our simulation study.

2 Motion Detection with Pixel-wise Measurement

2.1 Pixel-wise motion detection using Gaussian mixture model

The intensity history for a pixel from time 0 to time $t$ can be represented as a vector $\mathbf{x}_t = [x(0)\ x(1)\ ...\ x(t)]^T$ as shown in figure 1a). Here, we model this vector as a random vector drawn from a mixture of Gaussian distributions [18, 20]. Using $K$ distributions, the probability density function (pdf) of $\mathbf{x}_t$
can be expressed as
\[ p(x_t) = \sum_{i=1}^{K} w_{i,t} \ast p(x_t, \mu_{i,t}, \Sigma_{i,t}) \quad \text{with} \quad \sum_{i=1}^{K} w_{i,t} = 1, \]  
(1)

where \( w_{i,t}, \mu_{i,t}, \) and \( \Sigma_{i,t} \) are the estimates of the weight, the mean value vector, and the covariance matrix of the \( i^{th} \) Gaussian in the mixture, respectively. The \( i^{th} \) Gaussian pdf \( p(x_t, \mu_{i,t}, \Sigma_{i,t}) \) is defined as
\[ p(x_t, \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_{i,t}|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_t-\mu_{i,t})^T \Sigma_{i,t}^{-1}(x_t-\mu_{i,t})}, \]  
(2)

where \( \Sigma_{i,t} \) is assumed to be \( \Sigma_{i,t} = \sigma_{i,t}^2 I \) \[18\]. The \( K \) components in equation (1) are sorted in descending order based on \( \frac{w_{i,t}}{\sigma_{i,t}} \) such that \( \frac{w_{1,t}}{\sigma_{1,t}} \geq \frac{w_{2,t}}{\sigma_{2,t}} \geq \cdots \geq \frac{w_{K,t}}{\sigma_{K,t}} \). The weight to standard deviation ratio \( \frac{w_{i,t}}{\sigma_{i,t}} \) is a reasonable parameter to sort the components, because \( w_{i,t} \) represents the contribution of the \( i^{th} \) pdf to the mixture and variance estimate \( \sigma_{i,t} \) represents the uncertainty of the contribution. Using equation (1), we can determine if a new pixel measurement belongs to the current model. For example, we use the criteria that: at time \( t+1 \), \( x_{t+1} = [x(1) \ x(2) ... x(t+1)]^T \) belongs to the model if it is within 2.5 standard deviations of the mean value of any of the \( K \) Gaussian distributions.

The \( K \) distributions are divided into two groups of size \( B \) and \( K-B \). The first \( B \) distributions in equation (1) (i.e., those with the largest \( \frac{w_{i,t}}{\sigma_{i,t}} \) values) define a mixture model for a pixel without motion which defines a background pixel. A pixel at a position with motion is called a foreground pixel. For a new pixel measurement at time \( t+1 \), if \( x_{t+1} \) matches with any of the first \( B \) distributions, the pixel is a background pixel, otherwise a foreground pixel. \( B \) is defined as
\[ B = \min_b \left( \sum_{i=1}^{b} w_{i,t} > W \right), \]  
(3)

where \( W \) is a pre-defined threshold. Because the distributions in equation (1)
are sorted in descending order based on $\frac{w_{i,t}}{\sigma_{i,t}}$, the first $B$ distributions have the largest $\frac{w_{i,t}}{\sigma_{i,t}}$ values. When measurements at one pixel position in several frames belongs to one Gaussian distribution, the weight parameter $w_{i,t}$ is large. A small $\sigma_{i,t}$ means the measurements belong to the distribution with high certainty. Therefore the first $B$ distributions form a model for a background pixel, or a pixel with stable measurements. Note that the last $K-B$ distributions are affected by pixel measurements from last few frames. These measurements represent foreground pixels which might progressively represent background later.

After motion detection at time $t$, the measurement $x_{t+1}$ is used to update the mixture model in equation (1) with following online expectation maximization (EM) algorithm [20]:

\begin{align*}
\text{(1)} & \quad \hat{d}(w_{i,t}|x_{t+1}) = \begin{cases} 
1; & \text{if the } i^{th} \text{ Gaussian is the first distribution which } x_{t+1} \text{ belongs to} \\
0; & \text{otherwise}
\end{cases} \\
\text{(2)} & \quad \hat{w}_{i,t+1} = \hat{w}_{i,t} + \alpha \left( \hat{d}(w_{i,t}|x_{t+1}) - \hat{w}_{i,t} \right) \\
\text{(3)} & \quad \hat{\mu}_{i,t+1} = \hat{\mu}_{i,t} + \alpha \left( \frac{d(w_{i,t}|x_{t+1}) x_{t+1}}{\hat{w}_{i,t+1}} - \hat{\mu}_{i,t} \right) \\
\text{(4)} & \quad \hat{\Sigma}_{i,t+1} = \hat{\Sigma}_{i,t} + \alpha \left( \frac{d(w_{i,t}|x_{t+1}) \Delta \hat{\Sigma}_{i,t}}{\hat{w}_{i,t+1}} - \hat{\Sigma}_{i,t} \right)
\end{align*}

where $\alpha$ is an positive update parameter much smaller than 1, and $\Delta \hat{\Sigma}_{i,t}$ is a diagonal matrix which has same diagonal elements as the matrix $(x_{t+1} - \hat{\mu}_{i,t})(x_{t+1} - \hat{\mu}_{i,t})^T$. Note that $\hat{w}_{i,t+1}$, $\hat{\mu}_{i,t+1}$, and $\hat{\Sigma}_{i,t+1}$ can be rewritten as follows:

- $\hat{w}_{i,t+1} = (1 - \alpha) \hat{w}_{i,t} + \alpha \hat{d}(w_{i,t}|x_{t+1})$
- $\hat{\mu}_{i,t+1} = (1 - \alpha) \hat{\mu}_{i,t} + \alpha \frac{d(w_{i,t}|x_{t+1}) x_{t+1}}{\hat{w}_{i,t+1}}$
- $\hat{\Sigma}_{i,t+1} = (1 - \alpha) \hat{\Sigma}_{i,t} + \alpha \frac{d(w_{i,t}|x_{t+1}) \Delta \hat{\Sigma}_{i,t}}{\hat{w}_{i,t+1}}$

The parameter $\alpha$ is positive and $w_{i,t}$ is between 0 and 1, therefore observe that
the estimate of $\hat{w}_{i,t+1}$, $\hat{\mu}_{i,t+1}$, and $\hat{\Sigma}_{i,t+1}$ will be positive after each update. If $x_{t+1}$ matches none of the $K$ distributions, the last distribution in the model is replaced by a new Gaussian distribution with $x_{t+1}$ as its mean and an initially high variance and low prior weight in equation (1).

We refer to this Gaussian mixture model as the pixel-wise Gaussian mixture model. Foreground pixel detection is applied to each pixel position in a frame which also involves updating the mixture model for each pixel. Therefore, this pixel-wise motion detection method is computationally intensive. The memory and computational cost for the method are $O(KNt^2)$ and $O(KNt)$ respectively when there are $N$ pixels in the object scene. As discussed in introduction section, motion detection at each pixel is often used to search for a moving object. Finite moving object size suggests that detecting motion in an object block or a region over the whole object scene as opposed to doing it pixel-wise may be more appropriate. To solve this motion detection problem for an object block, we extend the pixel-wise Gaussian mixture model to a block-wise mixture model.

2.2 Block-wise motion detection using Gaussian mixture model

As shown in figure 1 b), pixel values within a block of size $\sqrt{N} \times \sqrt{N}$, at time $t$, are rearranged into a vector $x_b(t)$ of size $(N \times 1)$. The block pixels intensity history from time 0 to $L$ is represented as $x_{bL} = [x_b^T(0) \ x_b^T(1) \ ... \ x_b^T(L)]^T$. To simplify notation, $x_{bL}$ will be written simply as $x$. Note that the pixel-wise measurement vector $x_t$ consists of data points along a line in figure 1 a), while the measurement vector $x$ consists of data points in a cube in figure 1 b). Once again we model $x$ as a mixture of Gaussian
distributions similar to the model used for $x_t$. We refer to this model as a block-wise Gaussian mixture model. Using the same procedure as the one described in last subsection, a block vector measurement matching with any of the most significant $B$ Gaussian components in the model, is classified as a background block, otherwise it is declared as a foreground block. After making the motion detection decision, the block-wise Gaussian mixture model is updated with the new block measurements.

The block-wise motion detection method uses information jointly among neighboring pixels. Therefore, its theoretical performance should be no worse than the pixel-wise motion detection method. However, both detection strategies involve processing all pixels directly for updating the mixture models and making detection decisions. Also note that the decision for motion detection is binary. Therefore, we believe there is inherent redundancy in the measurement data for the binary decision problem. This motivates our compressive method in the following subsection, where the measurement vector $x$ is compressed into a lower dimensional feature space before making motion detection decision.

2.3 Feature-space motion detection using Gaussian mixture model

A feature of an object is defined as the inner product between an object $x$ and a projection vector from the feature basis. It can be represented as $y = Fx$, where $y$ is the feature vector of size $(M \times 1)$, $F$ is the projection matrix of size $M \times NL$, and $x$ is the object vector of size $NL \times 1$. With detector noise $n$, of size $N \times 1$ considered, feature representation is modified as $y = F(x + n)$. Here, we will consider 2D-Gabor [21], Hadamard binary, and random binary features for
the motion detection task. Figure 2 shows examples of the types of projections. Gabor and Hadamard projections are widely used in image processing, object detection and pattern recognition applications [22, 23, 24, 25, 26, 27, 28]. We consider binary non-negative Hadamard projection in this paper. Random projection has become popular in compressive sensing and machine learning field [29, 30]. We use binary non-negative random projections to ensure a fair comparison with binary non-negative Hadamard projections. Once again a Gaussian mixture model is used to detect motion in feature space.

Note that the choice of a particular feature is key in determining the motion detection algorithm performance. Therefore, features need to be sorted to optimize the algorithm performance and achieve the best performance with the least number of features. A divergence based feature selection method is used for sorting features [31]. In this method, a feature vector belongs to one of the two classes, class \( \omega_1 \) and class \( \omega_2 \) for blocks with and without motion, respectively. In each class, a feature vector is modeled as a random vector with a parametric model [31] constructed using training data. With this model, Kullback J-divergence between classes \( \omega_1 \) and \( \omega_2 \) can be measured and used to sort features. The J-divergence, is defined as

\[
J(\omega_1, \omega_2) = E_{y|\omega_1} \{ \log \frac{p(y|\omega_1)}{p(y|\omega_2)} \} + E_{y|\omega_2} \{ \log \frac{p(y|\omega_2)}{p(y|\omega_1)} \},
\]

where \( E_{y|\omega_i} \) denotes the mathematical expectation with respect to the class-conditional PDF \( p(y|\omega_i) \) with \( i = 1, 2 \). Observe that if the two classes are well separated, then \( \frac{p(y|\omega_2)}{p(y|\omega_1)} \) or \( \frac{p(y|\omega_1)}{p(y|\omega_2)} \) is large, resulting in a higher \( J(\omega_1, \omega_2) \) value. J-divergence [31], \( J(\omega_1, \omega_2) \) can be decomposed into independent components attributed to different features. Using an EM algorithm described in [31], the independent J-divergence components are computed, then the most
3 Motion Detection Performance Using Pixel-wise Measurements

A video clip with a hundred frames taken from the CAVIAR [32] database is used in the simulation study here. The video is filmed in the entrance lobby of the INRIA Labs at Grenoble, France. Each frame has size $288 \times 384$ with dynamic range $[0, 255]$. Figure 3 shows two example frames. To generate the ground truth, video frames are labeled by isolating the foreground and background regions manually. The object block size is chosen as $32 \times 32$ in this section. Four different noise levels $\sigma_0 = 0, 20, 100, 200$ are considered. The acronyms for the motion detection algorithms studied in this section are defined as follows:

(1) CP (Conventional measurement with Pixel-wise mixture model) algorithm: In this algorithm, Gaussian mixture model is employed for each pixel and foreground pixels are detected at individual pixel position. A block is decided as a moving block if there are $\kappa$ foreground pixels detected within the block, where $\kappa$ is a predefined integer between 1 and $NL$.

(2) CB (Conventional measurement with Block-wise mixture model) algorithm: In this algorithm, moving object block is detected using block-wise Gaussian mixture model.

(3) CDB (Conventional measurement with mixture model using Block Difference) algorithm: In this algorithm, Gaussian mixture model is formulated for the difference between object blocks in two consecutive frames. The difference is used for motion detection in each block.
(4) CFG, CFHB, or CFRB (Conventional measurement with mixture model using Gabor, Hadamard-Binary, or Random Binary feature) algorithm: In this algorithm, a Gaussian mixture model is formulated in feature space using Gabor, Hadamard binary, and random binary features. Features are used for block motion detection.

To evaluate algorithm performance, we employ the ROC (Receiver Operating Characteristic) and the AUC (Area Under Curve) metrics. The x and y axes in a ROC curve are the false positive rate (FPR) and the true positive rate (TPR) respectively. FPR is defined as the probability of motion detection when there is actually no motion, while TPR is defined as the probability of motion detection when motion is present. Based on the ground truth generated manually, a block with one or more than one foreground pixels is defined as a moving block. In each ROC curve, the largest TPR value is generated for a fixed FPR by searching the parameter space spanned by $L$, $\Sigma_{i,0}$, and $W$ variables in the motion detection algorithm. In CP algorithm, the threshold number of foreground pixels in each block, $\kappa$, is also optimized. The optimal values are used in all data presented here. The AUC value is calculated by integrating the area under the ROC curve.

Table 1 lists the AUC values using the algorithms discussed with noiseless measurement. In CFG, CFHB, and CFRB algorithms, $M = 1$ feature is used for motion detection. It can be observed from table 1 that all algorithms have detection performance close to 1, where 1 indicates perfect performance. Therefore, we conclude that $M = 1$ noiseless feature is sufficient for detecting motion when the object block has dimension 1024 ($32 \times 32$). Note that, the computation cost for CFG, CFHB, and CFRB algorithms is also reduced due to using the Gaussian mixture model in feature space. The amount of memory
and computation steps required are \(O(KML^2)\) and \(O(KML)\) respectively, where \(M \ll N\).

Figure 4 a), b), and c) shows the ROC curves for all algorithms at noise level \(\sigma_0 = 20, 100,\) and \(200\) respectively. It is observed that CP algorithm has better performance than CB and CDB algorithms at all noise levels. In CDB algorithm, the consecutive frame feature measurements are subtracted from each other, therefore the noise is twice as much as it is in other algorithms. Therefore, CDB algorithm has the worst performance when noise is high as \(\sigma_0 = 100,\) or \(200\). As discussed earlier, ideally CB algorithm should have no worse performance as CP algorithm. However, the data dimensionality of the observation vector in CB algorithm is much higher than CP algorithm, therefore estimation of the parameters such as \(\Sigma\) and \(\mu\) require more samples.

Figure 5 shows the ROC curves for CP and CB algorithm when noise level \(\sigma_0 = 100\) and object block size is \(4 \times 4\) and \(32 \times 32\). It can be observed when the object block size is small, the CB algorithm has superior performance compared to the CP algorithm.

Among the algorithms using features for motion detection, the CFG algorithm has the worst performance. From figure 2 we can observe that most Gabor projection vector elements have values close to 0. Therefore, using Gabor projection, limited signal power is collected in each feature. In feature space, signal to noise ratio using Gabor projection is smaller compared to using Hadamard and random binary projections. Also note that CFHB and CFRB algorithms have similar performance. Figure 6 a) and b) show two projection vectors used in CFHB and CFRB algorithms. The Hadamard projection vector does not have significant spatial variation over a block. On the other hand, the random projection vector has a lot of variation over the \(32 \times 32\) object block.
area. Therefore, we expect that the random vector would be superior to the Hadamard vector as a result of increased sensitivity to fine motion details. However, the motion in the experiment video used in our study occurs mainly on a large scale as shown in figure 6 c) and d). Therefore, in our case both algorithms yield similar performances.

It is also observed that when noise is high such as \( \sigma_0 = 100 \) and 200, detection algorithms employing Hadamard and random binary features achieve a superior motion detection performance relative to CP, and CB algorithms.

We compare the measurement SNR in CFHB and CP algorithms as a representative example to explain this observation. In the CP algorithm, the signal power is \( S_{CP} = E\{||x||^2\} = E\{\sum_{i=1}^{NL} x_i^2\} \), where \( x_i \) is the object pixel value in \( L \) consecutive object blocks with \( 1 \leq i \leq NL \). The noise power is \( \varepsilon_{CP} = NL\sigma_0^2 \) as the detector noise is assumed to be independent from each other. In CFHB algorithm, the feature value is defined as \( \sum_{i=1}^{NL} f_i(x_i + n_i) \).

Signal power is written as \( E\{\sum_{i=1}^{NL} f_i(x_i)^2\} \). Because any Hadomard binary projection has half elements zeros and the other half ones, the signal power can be reformulated as \( S_{CFHB} = 0.5 \times E\{\sum_{i=1}^{NL} x_i^2\} + E\{\sum_{i,j=1; i \neq j}^{NL} f_i f_j x_i x_j\} = 0.5 \times S_{CP} + E\{\sum_{i,j=1; i \neq j}^{NL} f_i f_j x_i x_j\} \). Notice the second term in \( S_{CFHB} \) is strictly non-negative. Using the same derivation it can be found out that the noise power in CFHB algorithm is \( \varepsilon_{CFHB} = 0.5 \times NL\sigma_0^2 = 0.5 \times \varepsilon_{CP} \). Putting the signal and noise power values together, it is clear that the SNR value in CFHB is larger than it in CP algorithm. Hence CFHB algorithm has better performance. Note that this performance improvement for CFHB due to SNR advantage is more significant when detector noise \( \sigma_0 \) is larger. To further improve motion detection method performance, the measurement SNR needs to
be increased. Note that using pixel-wise measurements to calculate a feature value includes noise from $NL$ detectors in each feature value. By employing a FSI system, features can be measured directly. This implies that each feature measurement includes noise only at one detector. This motivates us to study FSI system [33, 34, 35] performance for the motion detection task.

4 Motion Detection with Feature-specific Imaging System

To directly measure features in a FSI system, an electro-optic modulation device is employed. Examples of such electro-optic modulation device include a spatial light modulator (SLM) such as liquid crystal panel and digital micro-mirror device (DMD). A FSI system has several advantages over a conventional imaging system including fewer number of detectors, simpler hardware, lower cost, and higher measurement SNR [33, 36]. A FSI system can be implemented via three distinct optical systems: sequential, parallel, and pipeline architectures [34]. In a sequential architecture, the measurements are acquired sequentially one after the other. The measurements in parallel and pipeline architectures are taken simultaneously or in one snapshot. Among the three architectures, pipeline architecture presents the best photon efficiency, but incurs the highest system complexity. Relatively, the parallel architecture has superior photon collecting efficiency compared to sequential architecture and retains a simple structure. Therefore, in this paper, the parallel architecture is considered in the simulation study. Figure 7 shows an example FSI system employing the parallel architecture via a lenslet array. In such a system, the aperture contains a lenslet array with $M$ elements of equal size. Following each sub-aperture, a simple mask together with a detector is used to measure
one feature. Note that as the measured object illumination is proportional to
the sub-aperture area, signal intensity at each detector is \( \frac{1}{M} \) of the original
object total intensity. Detector noise is once again modeled as independent
white Gaussian noise with standard deviation \( \sigma_0 \). Feature measurements using
the parallel architecture FSI system are therefore mathematically defined
as \( y = \frac{1}{M} F x + n \).

To detect motion in blocks, we employ the same Gaussian mixture model as the
one used in CB. Note that in the FSI system, there is only one detector noise
component per feature measurement. Therefore, the noise power in all motion
detection methods using FSI measurements is \( \sigma^2_0 \), which much smaller than
\( NL\sigma_0^2 \) and \( 0.5 \times NL\sigma_0^2 \) in CP and CFHB algorithms respectively. Therefore,
we expect that by using FSI it is possible to achieve significant improvements
in motion detection algorithm performance.

5 Motion Detection Performance Using Direct Feature Measurements

The FSI-based motion detection algorithms using Gabor, Hadamard, and ran-
dom binary features are denoted by FG, FHB, and FRB respectively. Figure 8
shows the ROC curves for the three algorithms using \( M = 1 \) feature for three
detector noise levels \( \sigma_0 = 20, 100, \) and 200. Features are chosen according to
the J-divergence based feature selection method discussed in section 2. The
results for CP, CFG, CFHB, and CFRB algorithms are also presented for com-
parison. There are several observations that can be made from this figure. The
first observation is that FSI-based motion detection algorithms FG, FHB, and
FRB have significantly higher performance compared with the CFG, CFHB,
and CFRB algorithms, respectively. For example, when FPR has value 0.1 and noise level is $\sigma_0 = 20$, CFG, CFHB, and CFRB algorithms have TPR of 0.27, 0.93, and 0.91 respectively, while FG, FHB, and FRB algorithms have TPR of 0.83, 0.95, and 0.99, respectively. As the measurement SNR in FSI system is much higher than a conventional system, the motion detection algorithms using FSI system achieve better performance. The second observation is that FHB and FRB algorithms perform much better than FG algorithm. When FPR value is 0.1 with the noise level $\sigma_0 = 200$, the TPR values for FG, FHB, and FRB algorithms are 0.12, 0.96, and 0.94 respectively. As the Gabor projection vector has more than half elements with a value close to zero, the object energy collected in feature measurement is smaller than the energy collected in the Hadamard and random feature values. Comparing Hadamard binary and random binary projections, the performances for FHB and FRB algorithms are very similar. The third observation is that CP algorithm has better performance than FG algorithm, but much worse than FHB and FRB methods, especially at high noise levels. From this observation, we conclude that not all features in FSI system are superior to the conventional pixel-wise measurements for motion detection especially for low measurement SNR.

As shown in figure 9, FG algorithm has clearly worse motion detection performance through all three noise levels using $M = 4$ compared with $M = 1$ features. The same trend can be observed for the FHB and FRB algorithms with noise levels $\sigma_0 = 100$, and 200. This is due to the fact that the measured signal power is reduced by factor of four when $M = 4$ features are collected in the parallel architecture FSI system. As a result, the algorithm performance suffers from smaller measurement SNR. It is also important to em-
phasize that the projection vectors used in these algorithms are not
ordered/sorted according to their corresponding feature energy. In-
stead, as discussed in previous section, the projection vectors are
sorted based on the J-divergence distance between blocks with and
without motion in a feature subspace spanned by these vectors. Fig-
ure 10 shows the first 4 vectors of Gabor, Hadamard binary, and
random binary projections used in the simulation study ordered
(from left to right, left with the highest metric value) according
to our task-optimal metric. Accordingly, here we note that first
Hadamard binary projection vector is not the all-one vector which
would correspond to highest feature energy. For the purpose of
projection vector sorting we have utilized a set of training frames,
however, a different set of frames are used for testing and quanti-
fying motion detection performance to ensure that our results are
robust.

Figure 11 a) & b) show the ROC curves for FHB, and FRB algorithms using
1 feature measurement with different block size and two noise levels. Object
block with size 8 × 8, 16 × 16, 32 × 32, and 48 × 64 are considered. The
noise levels are σ₀ = 20 and 200. From both figures, it can be observed that
increasing the block size improves the performance, as a result of increasing
measurement SNR. The other observation is that system performance in FHB
and FRB algorithms do not degrade significantly when noise level increases
from σ₀ = 20 to σ₀ = 200. Thus, both the algorithms have robust performance
with increasing detector noise. This observation can also be supported by AUC
values for different detection algorithms as shown in figure 12 and table 2.

So far all results presented was on a training data set. To validate the per-
formance of the various algorithms, we now consider a test data set which is distinct from the training data set. Figure 13 presents a set of frames using CFHB and FHB algorithms. The area which has positive response for motion detection is indicated with higher intensity. The scene in this sequence of video frames is a car moving towards a building. Note that the tree leaves shaking due to wind are not a target of interest. As such, we consider them as clutter and would like them to be classified as background. The covariance matrix $\Sigma_{i,t}$ in the mixture model is assigned with large initial values to include the potentially moving clutter in background. From figure 13 it is clear that using the FHB algorithm the false alarm area is much smaller than using the CFHB algorithm. This demonstrates that motion detection using FSI system also works for scenes with clutter.

6 Conclusion

In this paper, we use the Gaussian mixture model developed by Grimson et al. [18] for block-wise motion detection. With pixel-wise measurements, several motion detection algorithms, CP, CB, CDB, CFG, CFHB, and CFRB, are studied. The CP algorithm is used as a baseline throughout the study. The Gaussian mixture model was extended to a block-wise mixture model, and the CB and CDB methods were considered. Then feature calculated from pixel values were considered for motion detection. It was observed that as a result of improved SNR in case of in CFHB and CFRB algorithms, they achieved larger AUC values than CP and CB algorithms, especially when detector noise is high. For example, compared with the AUC value 0.6269 for CP algorithm with noise $\sigma_0 = 200$, CFHB and CFRB algorithms have larger
AUC values 0.8108 and 0.7992 respectively. To improve the SNR further we considered FSI based feature measurements. Gabor, Hadamard binary and random binary features are collected directly as measurements. FHB and FRB algorithms show significant performance improvements comparing to the pixel-based feature motion detection algorithms CFHB and CFRB. However, the FG algorithm does not yield impressive performance because limited object energy is collected into feature measurement as the majority of Gabor vector elements have values close to zero. Therefore, although motion detection methods using FSI system have much better performance than using traditional system a proper choice of projection is critical to maintain a high measurement SNR.

In conclusion, motion detection methods using M=1 FSI system Hadamard and random binary feature measurement present higher than 0.95 AUC values over all noise levels $\sigma_0 = 0, 20, 100, \text{and } 200$.

For future work, we would like to consider different applications such as object motion detection and/or tracking in-vivo bio-imaging [37, 38, 39, 40]. We would also like to pursue an approach where the block size used for motion detection problem is determined adaptively based on the instantaneous needs of the imager. Here the expectation would be that the optimal block size would depend upon the spatial scale of the moving objects in a scene. In fact a spatially varying block size could further adapt to the different regions of the scene which may contain objects of different scales.
References


on particle filters for online nonlinear/non-gaussian bayesian tracking.


[40] T.W. Nattkemper. Automatic segmentation of digital micrographs: A
\[ x_t = [x(0) \ x(1) \ \ldots \ x(t)]^T \]

(a)

\[ x_{AL} = [x_b^T(0) \ x_b^T(1) \ \ldots \ x_b^T(L)]^T \]

(b)

Fig. 1. Object vector definition using a) one pixel; b) a block of pixels measurement.
Fig. 2. Examples for a) Gabor; b) Hadamard binary; c) Random binary projections.
Fig. 3. Examples for video frames a) without; b) with moving blocks.
Table 1
Area under ROC curves for motion detection algorithms using noiseless measurements

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>CP</th>
<th>CB</th>
<th>CDB</th>
<th>CFG</th>
<th>CFHB</th>
<th>CFRB</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>0.9985</td>
<td>0.9994</td>
<td>0.9991</td>
<td>0.9990</td>
<td>0.9994</td>
<td>0.9994</td>
</tr>
</tbody>
</table>

28
Fig. 4. ROC for algorithms using pixel-wise measurements for $\sigma_0$ is a) 20; b) 100; and c) 200.
Fig. 5. ROC for CP and CB algorithms using pixel-wise measurements for $\sigma_0 = 100$
with block size $32 \times 32$ and $4 \times 4$
Fig. 6. The first vector in sorted a) Hadamard; b) Random binary projections; c)d) Two examples of the difference between two consecutive frames. The grid shows the blocks area in a frame.
Fig. 7. FSI system with parallel architecture.
Fig. 8. ROC for algorithms using direct feature and pixel-wise measurements for $\sigma_0$

is a) 20; b) 100; and c) 200.
Fig. 9. ROC for algorithms using direct feature with $M = 1$ and 4 for $\sigma_0$ is a) 20; b) 100; and c) 200. The top part of each figure is magnified and presented for clarification.
Fig. 10. The first 4 projection vectors in Gabor, Hadamard Binary, and Random Binary projections ordered from left to right with left project being the most significant.
Fig. 11. ROC for CP algorithm, FHB and FRB algorithms using block size $8 \times 8$, $16 \times 16$, $32 \times 32$, and $48 \times 64$ with $M = 1$ for $\sigma_0$ is a) 20; b) 200. The top part of each figure is magnified and presented for clarification.
Fig. 12. Area under ROC curves for motion detection algorithms with block size $32 \times 32$. 
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>noise $\sigma_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>CP</td>
<td>0.9985</td>
</tr>
<tr>
<td>CB</td>
<td>0.9994</td>
</tr>
<tr>
<td>CDB</td>
<td>0.9991</td>
</tr>
<tr>
<td>CFG</td>
<td>0.9990</td>
</tr>
<tr>
<td>CFHB</td>
<td>0.9994</td>
</tr>
<tr>
<td>CFRB</td>
<td>0.9994</td>
</tr>
<tr>
<td>FG</td>
<td>0.9961</td>
</tr>
<tr>
<td>FHB</td>
<td>0.9993</td>
</tr>
<tr>
<td>FRB</td>
<td>0.9985</td>
</tr>
</tbody>
</table>
Fig. 13. Three consecutive frames with motion detected area highlighted, using CFHB a)~c) and FHB d)~f) algorithms; noise STD - $\sigma_0 = 200$, block size - $32 \times 32$, 1 feature used for each block.