sustainability

MDPI

Article

GPU-Accelerated Foreground Segmentation and Labeling for Real-Time Video Surveillance

Wei Song [1,2,*], Yifei Tian [1], Simon Fong [3], Kyungeun Cho [4,*], Wei Wang [5] and Weiqiang Zhang [4]

1 Department of Digital Media Technology, North China University of Technology, Beijing 100144, China; tianyifei0000@sina.com
2 Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data, North China University of Technology, Beijing 100144, China
3 Department of Computer and Information Science, University of Macau, Macau, China; ccfong@umac.mo
4 Department of Multimedia Engineering, Dongguk University, Seoul 04620, Korea; zhangweiqiang285@yeah.net
5 Guangdong Electronic Industry Institute, Dongguan 523808, China; wangwei@gdeii.com.cn
* Correspondence: sw@ncut.edu.cn (W.S.); cke@dongguk.edu (K.C.); Tel.: +86-10-8880-1991 (W.S.); +82-10-9070-9342 (K.C.)

Academic Editor: James Park
Received: 30 May 2016; Accepted: 22 August 2016; Published: 29 September 2016

Abstract: Real-time and accurate background modeling is an important researching topic in the fields of remote monitoring and video surveillance. Meanwhile, effective foreground detection is a preliminary requirement and decision-making basis for sustainable energy management, especially in smart meters. The environment monitoring results provide a decision-making basis for energy-saving strategies. For real-time moving object detection in video, this paper applies a parallel computing technology to develop a feedback foreground–background segmentation method and a parallel connected component labeling (PCCL) algorithm. In the background modeling method, pixel-wise color histograms in graphics processing unit (GPU) memory is generated from sequential images. If a pixel color in the current image does not locate around the peaks of its histogram, it is segmented as a foreground pixel. From the foreground segmentation results, a PCCL algorithm is proposed to cluster the foreground pixels into several groups in order to distinguish separate blobs. Because the noisy spot and sparkle in the foreground segmentation results always contain a small quantity of pixels, the small blobs are removed as noise in order to refine the segmentation results. The proposed GPU-based image processing algorithms are implemented using the compute unified device architecture (CUDA) toolkit. The testing results show a significant enhancement in both speed and accuracy.

Keywords: feedback background modeling; connected component labeling; parallel computation; video surveillance; sustainable energy management

1. Introduction

In the sustainable energy-saving strategies, smart meters provide an effective power reduction support by controlling electricity with intelligent functions [1]. Figure 1 shows the architecture of a sustainable energy management system consisting of three modules, which are monitoring, ubiquitous controlling, and power management. The monitoring module aims to sense the environmental information, including temperature, humidity, foreground objects, and other datasets. These sensing datasets are converted into several condition variables for evaluating an environment situation, which are then transmitted to smart meters via a low-cost and wireless sensor network based on ZigBee communication protocol [2]. Based on the received condition variables, the ubiquitous

controlling module achieves an auto-controlling approach for energy management using smart meters. Meanwhile, the smart meters enable remote operation of switching and setting values through cloud computing and controlling. In the power management module, the power socket of an appliance is linked with a smart meter, whose controlling signals are able to control the connected appliance, such as changing temperature of air condition, turning on or turning off lights, heating water, and other activities.

In such a sustainable energy management system, the real-time foreground objects detection in the monitoring module is an important task for smart controlling [3]. For example, in a garage, when a moving vehicle appears, the nearby lightings need to be turned on; if no foreground object exists, the lightings should be turned off so as to save electricity. To provide an effective environment monitoring approach, this paper aims to study a real-time and accurate foreground object detection algorithm for sustainable energy management. Meanwhile, fast and accurate foreground segmentation for video surveillance is a challenging task in image processing and computer vision [4]. This approach is also necessary for many multimedia applications, such as virtual reality, human-machine interaction, and mixed reality [5].

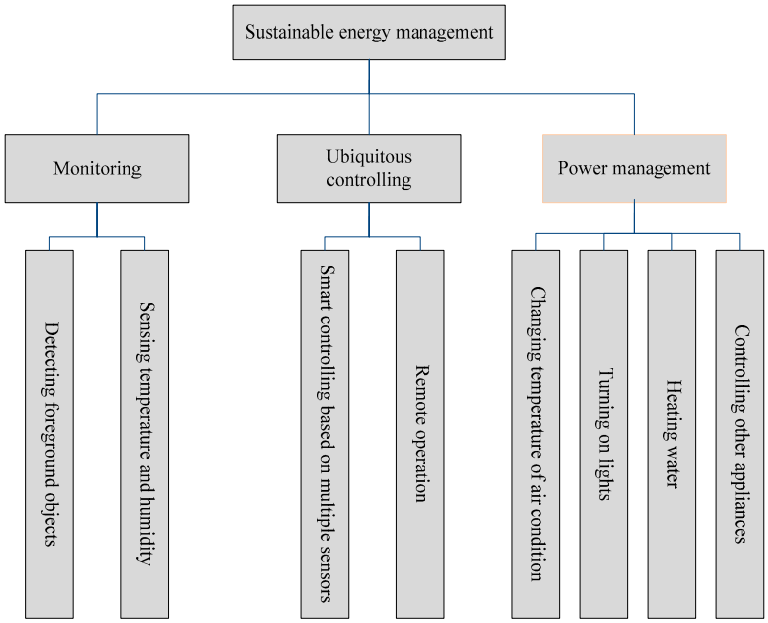


Figure 1. The framework of the typical sustainable energy management system.

Precise background subtraction is an important step for accurate foreground segmentation. Traditionally, a background model is initialized from the sequential images captured by stationary cameras [6]. The foreground pixels are subtracted with a pixel-wise difference detection method between the current image and the generated background model. To improve the segmentation accuracy, the adaptive background modeling methods are widely researched for situations where illumination is gradually changing, such as the codebook model [7]. Because the image scanning process causes huge computation, the traditional background model, such as codebook model, is initialized in the training phase, but not updated with the segmentation process simultaneously.

Although simple spatiotemporal background modeling strategies enable foreground detection without prior knowledgebase, noise and hole regions always exist in the segmentation results [8]. To enhance the foreground segmentation accuracy, some advanced technologies of background modeling algorithms are studied, such as adaptive background learning, Gaussian mixture model (GMM), self-organizing map, etc. [9]. However, it is difficult to segment foreground moving objects without noise in unconstrained outdoor environments in rainy, cloudy, foggy, and windy situations. Due to these weather issues, illumination variance of an air or object particle causes a short-term

noisy spot. Using the traditional algorithms, it is hard to determine whether the noisy spot belongs to background or foreground models.

The noisy spot is a blob of a small quantity of pixels in the segmented binary foreground result. Based on this definition, it is an effective solution to count pixels in each blob and remove the small blobs as noise. The connectivity-based clustering methods are widely researched to group the foreground pixels into several coherent blobs in spatial domain [10]. By labeling blobs with distinguished labels, we produce distinct objects for further processing steps such as objects tracking and analysis [11].

In the rough foreground segmentation process, the computation complexity of pixel-wise difference detection is so easy that foreground pixels can be subtracted fast using CPU programming. However, in the refinement process, the iterative scanning processes of the connected component detection, labeling and noise removal cause huge computation consumption. It is difficult to utilize the traditional CPU computation method for real-time connected component labeling (CCL). To further improve the processing speed, this paper proposes to utilize a graphics processing unit (GPU) programming method instead of CPU to speed up the feedback background modeling, foreground segmentation, and CCL algorithms.

Using GPU programming, we create a thread for each grid in order to analyze its local dataset in parallel [12]. In pixel-wise foreground segmentation algorithms, the image processing in each pixel is independent from others so that the traditional CPU-based algorithm is easy to be implemented [13]. For each pixel, we create a color histogram to record its color-changing information, which combines into a background model. If a pixel color in the current image does not locate around the peaks of its histogram, it is segmented as a foreground pixel. Meanwhile, the current image updates the background model synchronously.

In CCL algorithms, the image processing of a pixel is affected by its neighbor pixels [14]. When we apply parallel computation to process such dependent situations, a thread event on the neighbor grids keeps on being processed no matter that the processing results executed by the threads of the neighboring pixels [15]. Because the GPU threads do not process at the same speed, the parallel computing results are not certainly same for the dependent situations. For example, in CCL algorithm, we need to label a clique with a minimum value among them, while all labels update based on their neighboring labels simultaneously. If a thread of a neighbor pixel is faster than others, its label will be updated earlier among its local clique so as to cause uncertain labeling results. If the pixel thread is locked until its neighboring computations are finished, there is no difference between CPU and GPU processing methods. For CCL acceleration, we develop a novel parallel connected component labeling (PCCL) algorithm to speed up the image processing of the dependent situation. After the foreground pixels are clustered into several blobs, the pixel count of each blob is computed. The noise blob is assumed to have small quantity of pixels so as to be removed based on the PCCL results.

The main contributions of the proposed PCCL method consist of accurate foreground segmentation and real-time component labeling. The GPU-based feedback background modeling algorithm is able to update the background model of pixel-wise color histogram in real time. The PCCL algorithm increases the labeling speed for component labeling in high-resolution videos. Our proposed method is compatible with real-time remote monitoring, virtual-physical collaboration, sensor network, and other multimedia applications.

The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 introduces the proposed feedback background modeling method and the PCCL algorithm. Section 4 evaluates the performance of the proposed methods. Section 5 concludes the paper.

2. Related Works

Currently, feedback background modeling algorithms are widely researched for detection of moving objects. Cuevas et al. [16] presented a nonparametric background modeling strategy with a particle filter for lightweight smart camera applications. In this strategy, the probability density

functions based on Gaussian kernels were defined at each pixel to nonparametrically estimate whether it belonged to the image background or moving foreground regions. Ramasamy et al. [17] and Casares et al. [18] employed a temporal difference detection method to generate adaptive background models in dynamic scenes for surveillance enhancement. By detecting the difference between the consecutive frames, the foreground regions of interest were removed. The background model was modeled by the segmented background image frames. Azmat at al. [19] proposed a low-cost adaptive background modeling method by analyzing temporal scene events including observation frequency and longevity. Using such pixel-wise threshold and frame difference methods, there were noise and holes existing in the segmentation results of the complex scenes with cluttered objects.

Gallego et al. [20] proposed a foreground segmentation system with a tracking algorithm in camera capture videos. In this system, pixel-wise difference detection between current image and background model was implemented in the initialization phase. The background modeling method modified the mean-shift algorithm with a feedback background model, which enhances the foreground detection accuracy. Zamalieva et al. [21] introduced a background subtraction method in complicated scenes captured at dynamic camera viewports. The background geometric structure was constructed and the foreground pixels were detected from a series of one-direction conversion between the frames. In these methods, the feedback background modeling phases causes huge computation consumption for high-resolution video sequences.

Currently, the traditional data analysis algorithms are enhanced using GPU technologies [22], which enable parallel computation of large-scale datasets. The compute unified device architecture (CUDA) from Nvidia (Nvidia, Santa Clara, CA, USA) provides GPU programming interfaces, which is widely studied for pixel-wise background modeling and foreground segmentation [23]. To extend the typical CPU-based GMM background modeling method [24], Pham et al. [25] and Boghdady et al. [26] utilized CUDA libraries to implement the background modeling process in parallel. They created a pixel-wise probability distribution of intensity, which was updated by each allocated pixel thread in parallel. To remove shadow effect under intensity changes, Fukui et al. [27] recorded illumination intensity and RGB (red, green, blue) color information into a histogram table as an adaptive background model. The intensity and color threshold of each pixel were generated from their distribution stored in the corresponding histogram. The histogram buffer was allocated and updated in GPU memory so as to record each pixel's information in parallel. Although the background modeling and foreground segmenting processes were able to be implemented synchronously using these methods, noise and holes existed in the foreground segmentation results.

To realize real-time and accurate foreground–background segmentation, Griesser et al. [28] applied a pixel-shaders GPU programming technology to accelerate the iterative image scanning speed. The noise was removed and holes were filled in the rough segmentation results using an iterative Markov random field model. Cheng et al. [29] proposed an online learning framework for foreground segmentation to adapt for spatio and temporal changes of the background scene. For that their proposed algorithms were compatible with parallel computation formalism, they utilized a GPU programming to facilitate real-time foreground segmentation from dynamic scenes over time. These GPU-based foreground segmentation methods achieved real-time approaches, but the noise still existed in difficult scenarios such as foggy and maritime environments.

Connectivity-based clustering methods are studied for distinguishing and tracking objects so as to segment foreground without noisy speckle. Wang et al. [30] applied an adaptive connected component analysis method to remove the small spots of incorrect foreground pixels from the segmentation results of visual background extractor (ViBe). Using a single-chip Field Programmable Gate Array (FPGA) for the segmentation of moving objects in a video, Appiah et al. [31] developed a novel feedback background updating method with a CCL algorithm. The moving foreground segmentation and labeling processes were implemented in parallel on the FPGA hardware platform. Using a morphological opening method, the noise was removed and the holes were filled. Jiang et al. [32] developed connectivity-based segmentation in 2D/3D sensor networks for geometric environment understanding. Flooding from the boundary nodes, the whole geometric cells were scanned to

construct a Reeb graph. If the neighbor cells were not mutex, they were merged into one component. The iterative scanning for counting the blobs pixel by pixel causes huge computational consumption. To speed up the connectivity detection in binary images, He et al. [33] enhanced the conventional two-scan CCL algorithm. On the first scan, all labels of a connected component were assigned with the same temporary equivalent integer. On the second scan, the corresponding labels in a connected component were assigned their unique representative labels. For visual surveillance of moving objects, Hu et al. [34,35] proposed a fast 4-connected component labeling algorithm to reduce the influence of a dynamic environment, such as wave ripples. In each scan process of this algorithm, the adjacent pixels of two rows were merged into an isolated block, which was labeled with the minimum label value in the block. After the foreground pixels were roughly subtracted by an illumination difference detection method, the noise assumed as small isolated block was removed. These CPU-based component labeling algorithms were able to deal with the low-resolution video images, but hard to scan large foreground blobs in real time.

In contrast to these state-of-the-art sequential CCL methods, parallelism and multithreading strategies accelerated the labeling speed in high-resolution videos. Typically, there were two propagation methods implemented in the CCL kernel, including directional pass and multi-way pass. Using CUDA, Kalentev et al. [36] and Hashmi et al. [37] implemented iterative row–column scanning on 2D binary grids. After initializing all non-zero element labels with their corresponding index, this algorithm propagated the lowest label value along the rows and the columns to label all non-zero neighbors with this lowest value. Hawick et al. [38] imaplemented and analyzed the performance of two-pass, one-pass, and multi-pass CCL algorithms for speeding up graph component labeling. They allocated a thread for each pixel, which propagated its adjacent pixel list and labeled them with the minimum label value of them. The propagation iterations were stopped and the labeling process was finished until all labels were not changed. In the directional propagation labeling algorithms, such as one-pass, two-pass, and row–column scanning, a GPU thread scanned a row and a column at least twice. The first scanning was to search minimum label, and the second scanning was to label all grids in the row and column with the minimum label. This scanning method was effective for low-resolution images. In contrast to CPU thread, the processing speed of GPU thread was reduced when the computation complexity became high. Thus, the row–column scanning algorithm of CCL caused computational overload for high-resolution situations. From their experiments, the multi-pass algorithms provided the best performance. In description of their multi-pass algorithms, each thread searched and labeled the corresponding local and neighbor pixels synchronously. Figure 2 shows an example of two iterations of their method. To keep synchronous reading and updating labels, a GPU thread executed its function and held on until every thread finished searching the minimum label. Also, a GPU thread only labeled the local pixel with the minimum label, instead of propagating the minimum label to its neighbor pixels. Although this method kept the consistency of the CCL results, a large number of iterations were required for large-scale datasets due to the low bandwidth of the labeling process.

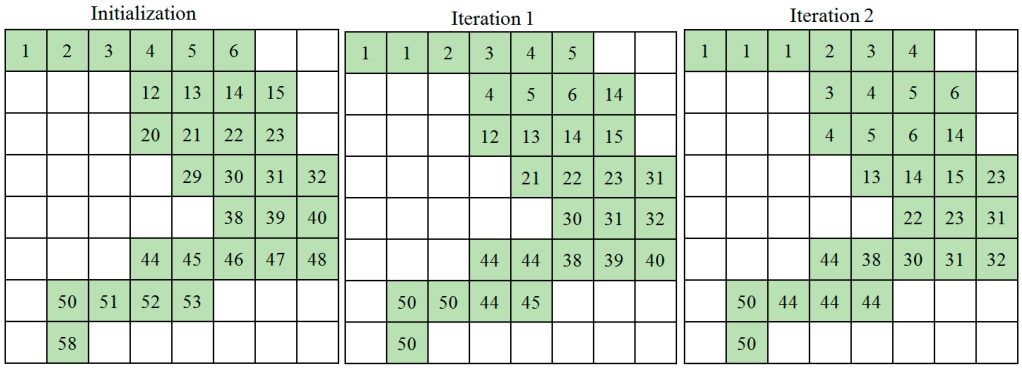


Figure 2. The multi-pass connected component labeling (CCL) algorithm proposed by Hawick et al. [38].

Using the threads synchronization function, the parallel computation speed was reduced, because the threads had to wait for all threads to finish. Meanwhile, the labeling process became slow without propagating the minimum label to neighbor pixels. If the GPU threads processed the neighbor propagation without synchronization, the data-dependent issue in Figure 3 would arise, because the GPU threads executed with different speeds. Taking the first four pixels as an example, the labeling process in thread 4 was slower than thread 2, so that the labeling result "1112" was covered by "1133". This asynchronous characteristic of GPU threads caused uncertain iteration times.

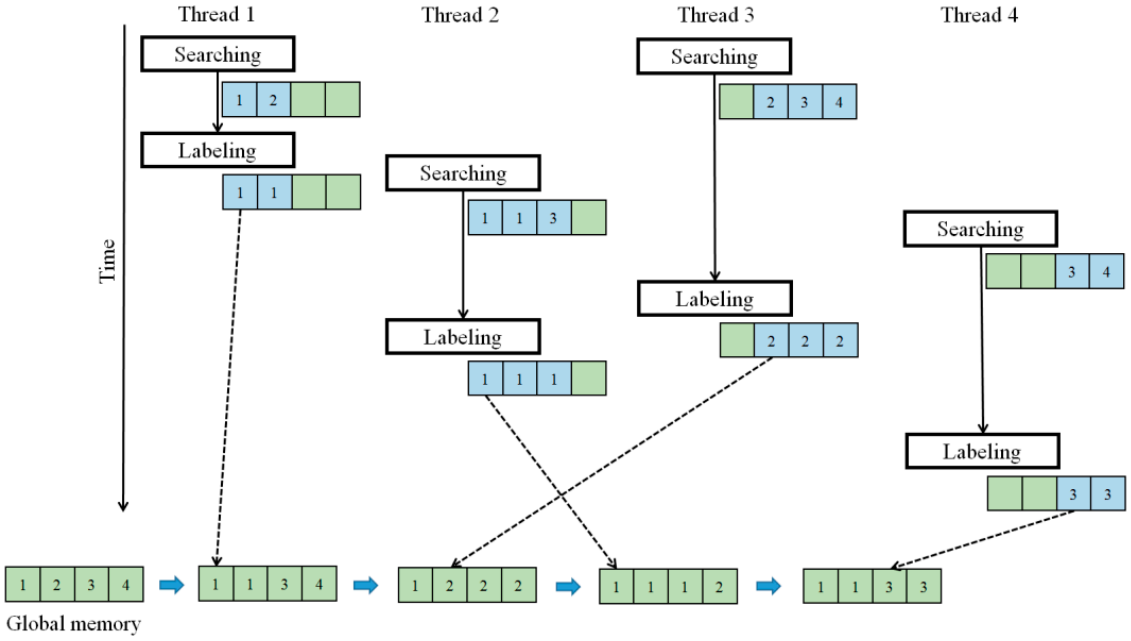


Figure 3. The data-dependent issue of the propagation process in the graphics processing unit (GPU)-based CCL algorithm.

The GPU-based CCL algorithms should address the data-dependent issues. Cabaret et al. [39] benchmarked the existing GPU-based CCL algorithms and concluded they were all multi-pass data-independent approaches, which were not suitable for GPU acceleration. Instead of GPU programming technology, they utilized OpenMP to program a parallel version of CCL on multi-core processors. However, such multithread architecture of CCL was still inefficient to deal with data-dependent scanning process in parallel. To realize the neighbor propagation in parallel, this paper presents a novel PCCL algorithm to increase the labeling speed for the data-dependent situations.

3. GPU-Accelerated Foreground Segmentation and Labeling

This section describes the GPU-accelerated foreground segmentation and labeling methods using a GPU programming technology. We propose a feedback background modeling method for foreground segmentation from video sequences. From foreground pixels, a PCCL algorithm is developed to cluster and label them into several distinguish blobs.

3.1. The GPU-Accelerated Framework

In contrast to CPU-based image processing methods, a GPU-accelerated framework for foreground CCL in video images is proposed as in Figure 4. The framework contains two modules, including foreground segmentation and labeling. Using CUDA software development kits (SDKs), we allocate $(W \times H - 1)/T + 1$ blocks and T threads in each block in order to create $W \times H$ threads to compute the proposed image processing algorithms on each pixel in parallel. Here, W and H are the width and height of the video image separately.

In the foreground segmentation module, we allocate the video image and the background model buffers in GPU memory. In the background model, pixel-wise RGB histograms are created for recording the color changing distribution. The background model keeps updating based on new registered video images. Meanwhile, the foreground pixels are segmented, if their colors do not locate around the peaks of their corresponding histograms.

In the CCL module, the labels of the segmented foreground pixels are initialized with their corresponding indices. After a GPU thread searches the minimum label among a pixel and its neighboring pixels, these pixels are assigned with the minimum label. This propagation process is implemented iteractively until there is no changing in all labels. This way, the segmented foreground pixels are clustered into several distinguished blobs.

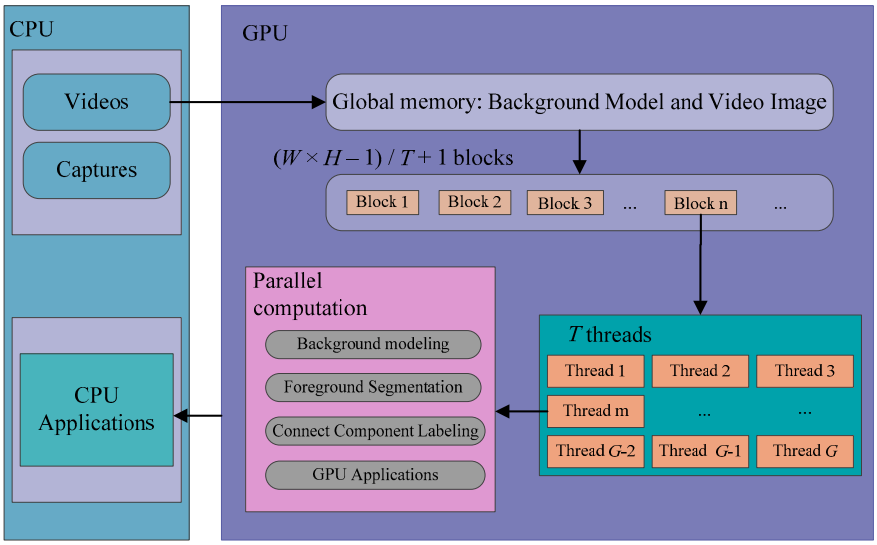


Figure 4. The GPU-accelerated framework for foreground segmentation and labeling.

3.2. *Foreground Segmentation by a Feedback Background Modeling Method*

This section proposes a feedback background model using GPU programming technology for foreground segmentation. As shown in Figure 5, this model implements the background modeling and segmentation processes synchronously.

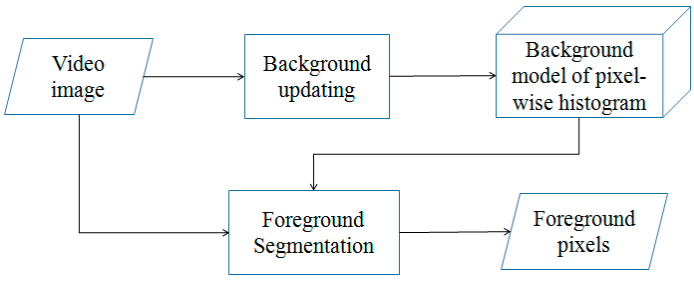


Figure 5. The dataflow of the feedback background modeling and foreground segmentation.

As shown in Figure 6, in GPU memory, we allocate pixel-wise RGB color histograms and background colors of the histograms for each pixel. To record the color changing history of red, green, and blue channels, $H \times W \times 3$ histograms are updating with the new registered color at the corresponding pixels. Each histogram contains 256 integer buckets to record the color appearance counts. When a color (r, g, b) at a pixel (x, y) is registered to the background model, the corresponding histogram M_i at the bucket i is updated as follows:

$$\begin{aligned} M_{Wy+x+HWr} &= M_{Wy+x+HWr} + 1 \\ M_{Wy+x+HWg+257HW} &= M_{Wy+x+HWg+257HW} + 1 \\ M_{Wy+x+HWb+2\times257HW} &= M_{Wy+x+HWb+2\times257HW} + 1 \end{aligned} \quad (1)$$

A pixel is classified to background class by determining whether its color is repeatedly appearing. During the training stage, the background colors are registered around the peaks in the pixel-wise histograms. From the histograms, the background colors corresponding to the peaks are stored in the 256th buffer blocks. During the testing stage, if a pixel color in a new registered image locates around the peaks of its RGB histograms, it is determined as a background pixel. Otherwise, this pixel is segmented as a foreground pixel. Using GPU programming, the background training and testing stages are implemented in parallel so as to realize a synchronization approach.

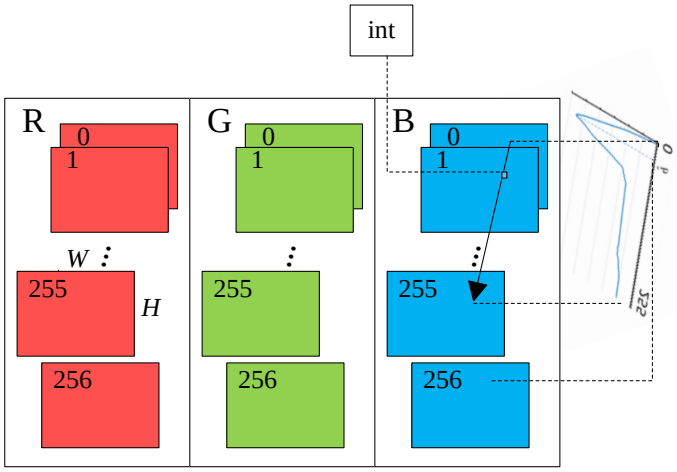


Figure 6. The background model of pixel-wise color histograms in GPU memory.

3.3. PCCL Algorithm

This section proposes a PCCL algorithm to cluster the segmented foreground pixels into several separate blobs using GPU programming. To realize a parallel computation approach, a GPU thread is created to process the labeling and propagation processes for each foreground pixel, as shown as the red color regions in Figure 7a. A thread has an ID equal to its processed pixel index.

A label map L in Figure 7b is initialized from the foreground segmentation result. If a pixel i is classified as a foreground pixel, the label at i is specified as $L(i) = i$; otherwise $L(i)$ = null. In the connectivity detection stage, each foreground pixel propagates to its adjacent foreground pixels which are labeled with the minimum label value among them. In the typically CCL algorithm, a pixel (x, y) propagates its right and bottom pixels, as shown in Figure 8a. In our proposed PCCL, we enlarge the propagation range for accelerating the convergence of the iterative scanning. As shown in Figure 8b,c, a neighboring clique for a pixel (x, y) is defined as the adjacent and local pixels to the pixel with a distance of d ($d \geq 1$).

We illustrate the propagation in Figure 7c with $d = 1$. In the CPU-based sequential computing methods, the propagation process is implemented pixel by pixel. In contrast to them, we process all foreground pixels individually in parallel by $H \times W$ threads. Because a foreground pixel and its neighboring foreground pixels propagate to their neighboring cliques synchronously, the computation of local and the neighboring labels are not certainly finished. Thus, the labeling result is not always the same. When this parallel propagation process is executed iteratively until all label values no longer change, the connected pixels are labeled with the lowest label value among them after several iteration operations, as shown in Figure 7d.

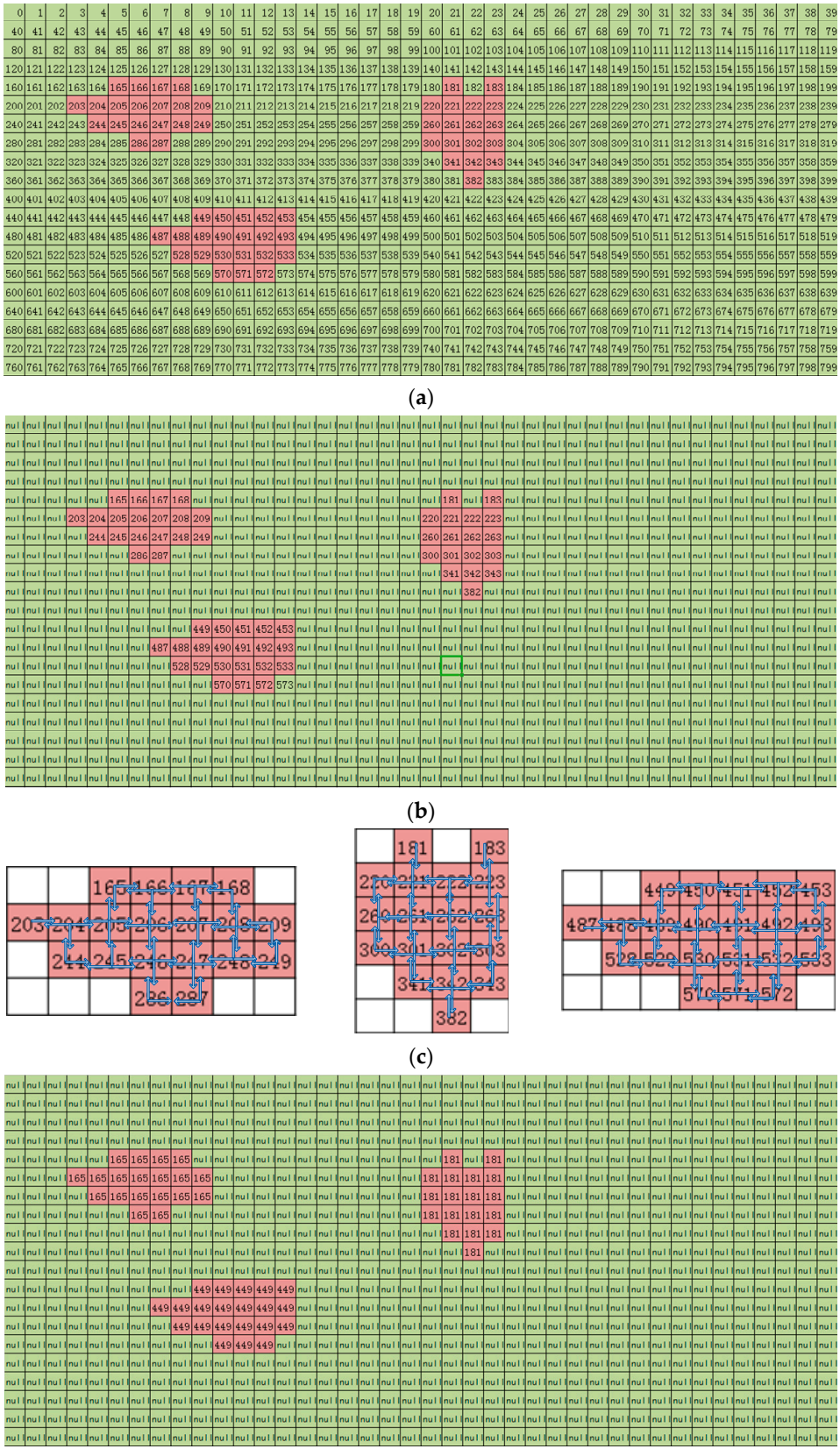


Figure 7. A processing example of the proposed parallel connected component labeling (PCCL) algorithm. (**a**) The segmented foreground pixels; (**b**) the initialized label map; (**c**) the propagation process from any foreground pixel to its neighboring clique; (**d**) the labeling result.

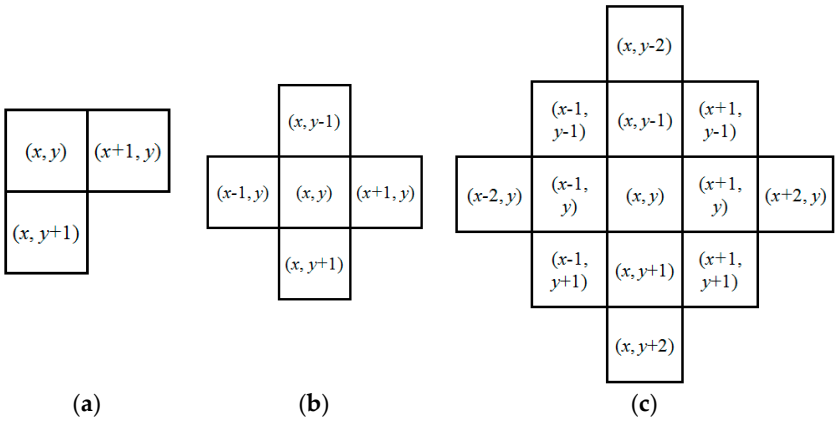


Figure 8. The propagating neighboring clique. (**a**) The typical clique; (**b**) the proposed clique of $d = 1$; (**c**) the proposed clique of $d = 2$.

3.4. Components Extraction

In object tracking applications, distinguished components need to be extracted respectively from the labeling result. Also using GPU programming, we propose a parallel components extraction method as shown in Figure 9.

A component label *cl* is defined as the value of its pixel labels, which is the minimum index among them. By traversing all labels using $H \times W$ threads, a *cl* is selected and valued as any label equal to the label index. Then, we extract the component pixels in the component *cl*, and label their labels as null. This process is implemented iteratively until any component label does not change in the label map.

Besides objects tracking applications, this method provides a clue to remove noise from the foreground segmentation. The noise is always formed as a blob containing a small quantity of pixels. If the extracted pixel of a component is small, we determine that this component is noise and it is removed from the segmentation results.

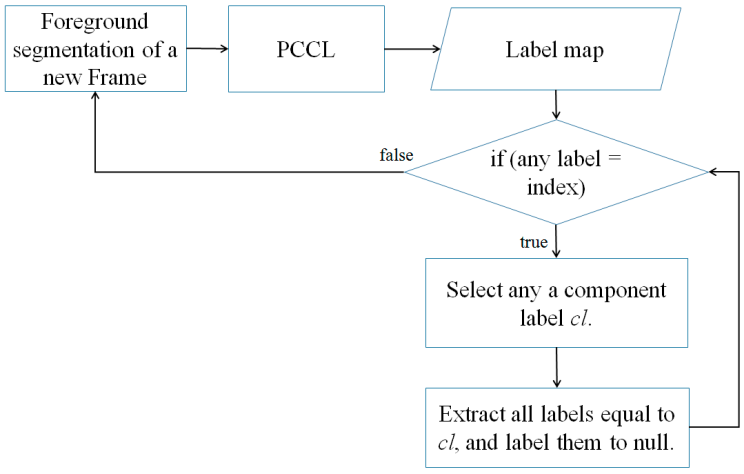


Figure 9. The dataflow of the components extraction from the labeling result computed by PCCL.

4. Experiments and Analysis

In this section, we analyze the performance of the proposed GPU-accelerated foreground segmentation and labeling method. The experiments were implemented on a 3.20 GHz Intel® Core™ (Intel, Santa Clara, CA, USA) Quad CPU computer with a GeForce GT 770 graphics card (Nvidia, Santa Clara, CA, USA), 4 GB RAM. The GPU-based parallel computation was implemented using the CUDA toolkit V5.5 (Nvidia, Santa Clara, CA, USA).

4.1. Foreground Segmentation Performance Analysis

Figure 10 shows the experimental results of the proposed methods, which were executed on several typical and challenging scenarios from a crowd dataset (PETS2009) [40], vehicle dataset (AVSS2007, London, UK) [41], jug dataset (ICCV2003, Nice, France) [42], and WaveTree dataset (Microsoft Research, Redmond, WA, USA) [43], from left to right. In these datasets, the camera viewports were stationary. The crowd dataset contained a 768 × 576 AVI video for the evaluation of moving pedestrian detection. The vehicle dataset had 720 × 576 video sequences for the evaluation of vehicles segmentation, which contained parked vehicles. The jug dataset contained a set of 320 × 240 sequential images for evaluating the moving semitransparent jug segmentation from maritime environments where wave ripples existed. The WaveTree was a low-resolution dataset of 160 × 120 sequential images, which had a waving tree as a background object. The available buffer size of the applied NVIDIA hardware was 5622 megabytes, which satisfied the GPU memory requirement of the background modeling for our experiments.

After several frames of background model updating, the background images in Figure 10b were generated from the color values located in the peaks of the color histograms. Then, we implemented the proposed PCCL algorithm in the foreground pixels segmented using the generated background model. The connected components were rendered with distinguished colors in the labeling results, shown in Figure 10c. After the components containing small quantity of pixels were removed as noise, the foreground segmentation results were refined and the foreground objects as large components were extracted, shown in Figure 10d. Finally, the foreground component pixels were extracted in Figure 10e by masking the binary map in Figure 10d to the original images.

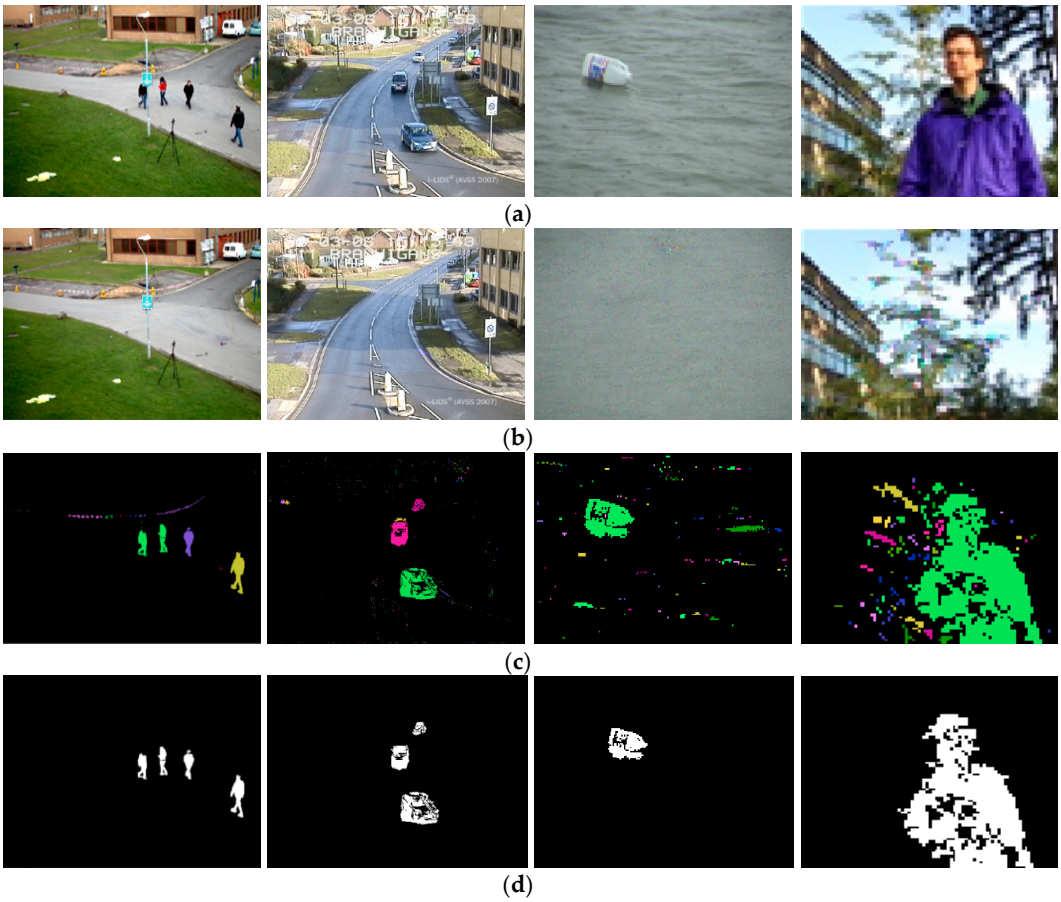
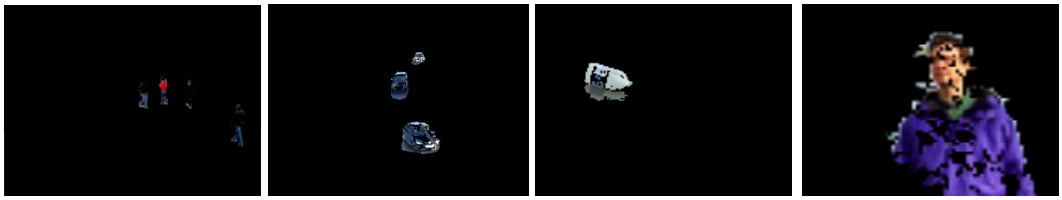


Figure 10. *Cont.*



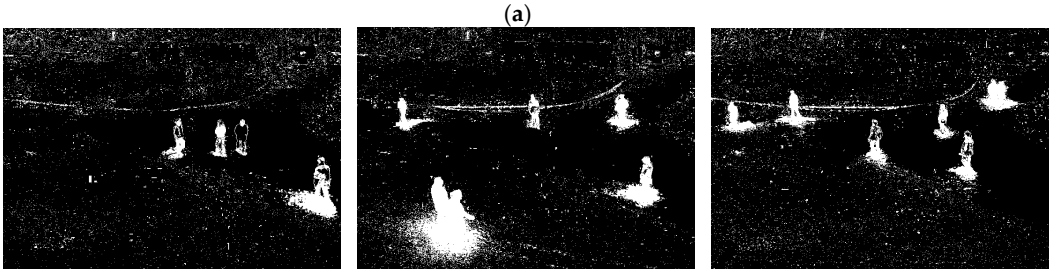
(**e**)

Figure 10. The experimental results of proposed foreground segmentation method using the PCCL algorithm. (**a**) The original video images; (**b**) background image generated from the histogram peaks in the background model; (**c**) the labeling results; (**d**) the foreground component extraction without noise; (**e**) the foreground pixel extraction results.

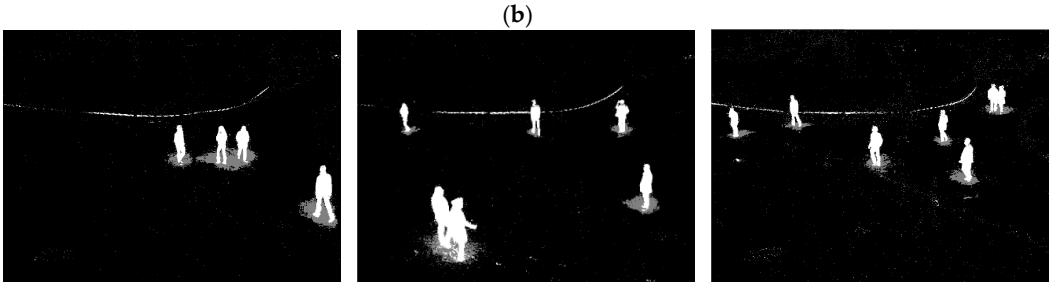
In order to analyze the performance of our proposed foreground segmentation and PCCL methods, we conducted the same experiments using a self-adaptive codebook model proposed by Shah et al. [44] and the typical GMM proposed by Stauffer and Grimson [24]. Following the codebook and GMM description in [24,44], we programmed the codebook and GMM using the OpenCV library (Itseez, San Francisco, CA, USA).

Figures 11 and 12 illustrate the qualitative comparison results of these foreground segmentation methods on the 768 × 576 crowd dataset and 640 × 480 campus dataset, respectively. Figures 11a and 12a have several sampling frames of the original videos. The foreground segmentation results using the codebook model are shown in Figures 11b and 12b, which contained the noise caused by illumination changes and small waving objects. Although the GMM model removed the noisy spot caused by varying illumination, the spattered noise of waving objects still existed in the segmentation results, as shown in Figures 11c and 12c. Using the PCCL algorithm, our proposed method removed both spot and spattered noise of a small quantity of pixels and provided accurate foreground detection results in Figures 11d and 12d.

(**a**)

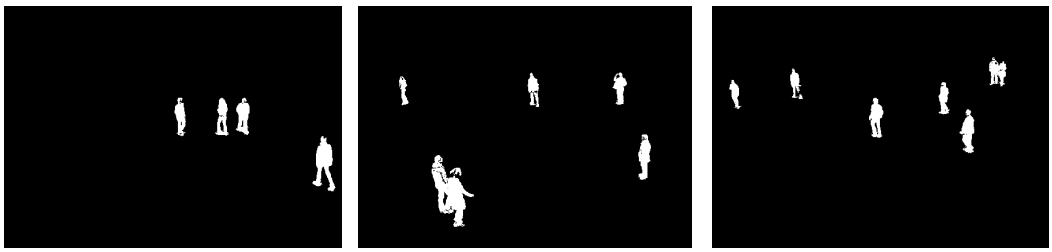


(**b**)



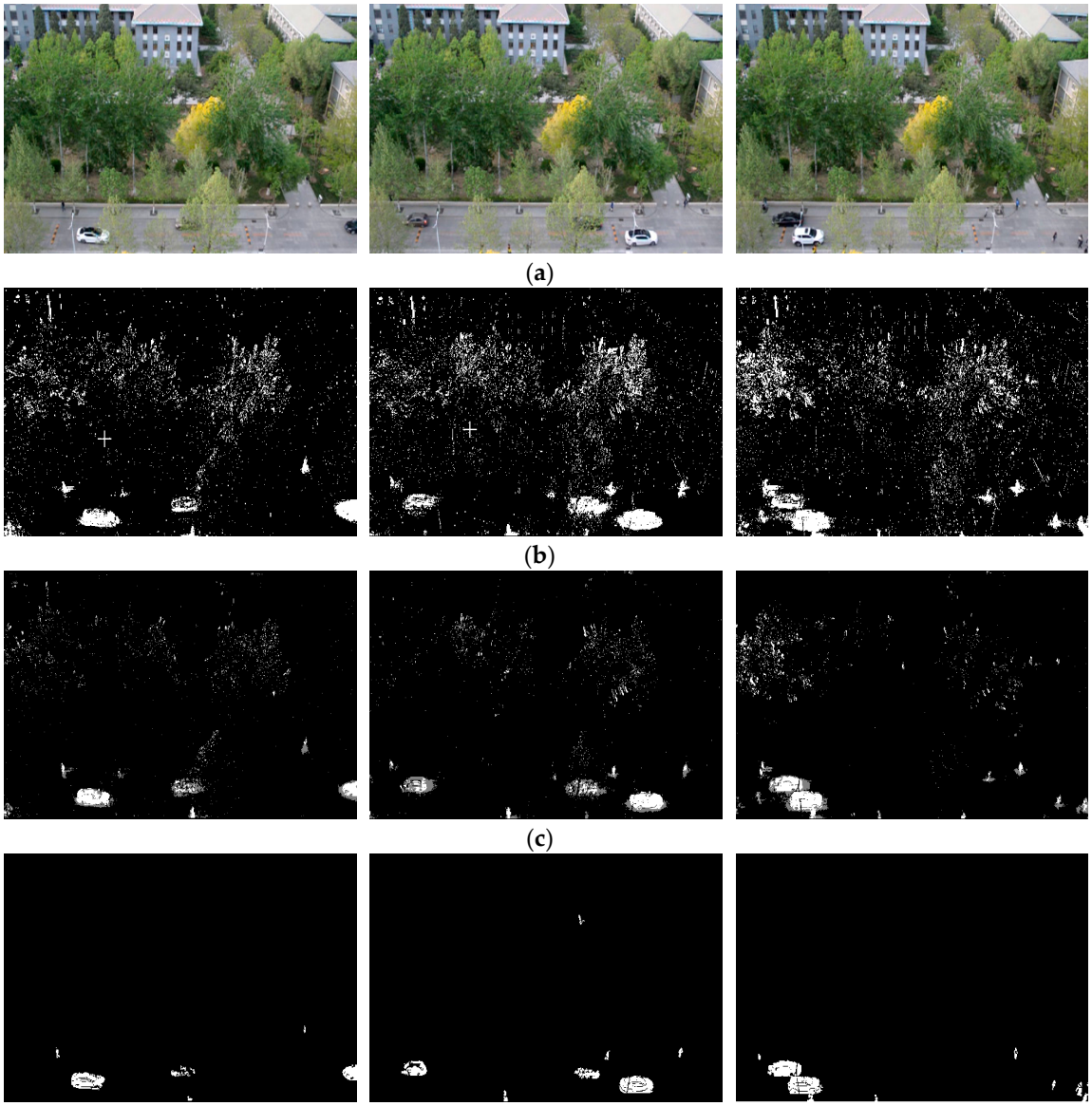
(**c**)

Figure 11. *Cont.*



(**d**)

Figure 11. The qualitative comparison results on the crowd dataset. (**a**) The original video images; (**b**) the segmentation results using the codebook model; (**c**) the segmentation results using the GMM; (**d**) the segmentation results using our proposed method.



(**a**)

(**b**)

(**c**)

(**d**)

Figure 12. The qualitative comparison results on our campus dataset. (**a**) The original video images; (**b**) the segmentation results using the codebook model; (**c**) the segmentation results using the GMM; (**d**) the segmentation results using our proposed method.

Figure 13 shows the speed performance of the foreground segmentation and labeling methods compared with the self-adaptive codebook model, the GMM, and the GPU GMM proposed by Pham et al. [25]. We implemented the training and testing stages synchronously. The average processing speeds of the foreground segmentation in the 640 × 480 resolution dataset were 39.3867 frames per second (fps), 6.8981 fps, 29.1489 fps, and 56.3287 fps by using the codebook model, the GMM, the GPU GMM, and the proposed feedback model with the PCCL, respectively. For the 768 × 576 resolution dataset, the processing speed reduced to 15.8867 fps, 4.6507 fps, 20.6014 fps, and 34.1488 fps. The results reflect that when the video resolution became high, the foreground segmentation using codebook and GMM models were not able to implement in real time. The GPU thread is fit for computing simple programs in parallel, but becomes slow when processing complex algorithms. The Gaussian probability density function computation was so complex that the GPU computation speed of GMM was not fast enough for real-time computation in high-resolution videos. Although the processing speed of our method was reduced in the high-resolution situation, it was more than 30 fps and satisfied the requirement of real-time monitoring.

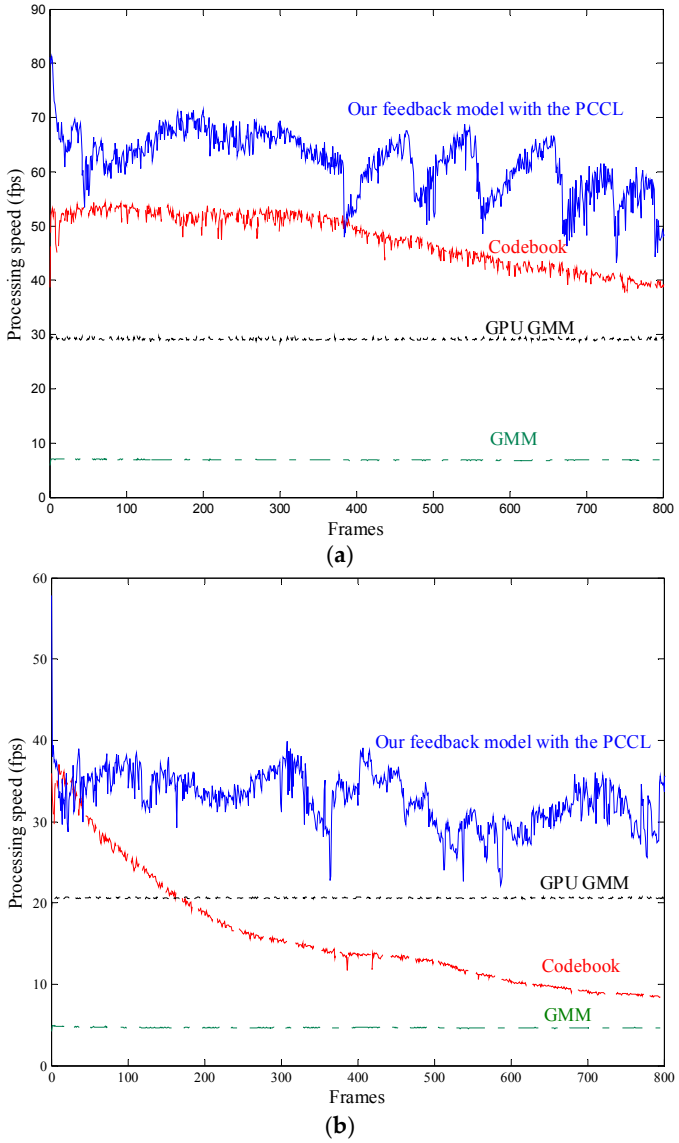


Figure 13. The foreground segmentation speed performances of the proposed feedback model with the PCCL, compared with the codebook, the GMM, and GPU GMM. (**a**) The dataset of 640 × 480 resolution; (**b**) the dataset of 768 × 576 resolution.

4.2. PCCL Performance Analysis

The existing CPU-based directional pass and multi-way pass CCL methods utilized a single thread to propagate the minimum label to the local and neighboring dataset pixel by pixel. Their principles were similar such as descriptions by He et al. [33] and Hu et al. [34]. The different performances of iteration and processing speeds were caused by the different label array range and testing different resolution. If the label array range was enlarged, the minimum label of the large range was quickly searched propagated to the large range by an execution. However, the processing speed became slower due to the more computation consumption required in the larger labeling range.

To compare the performances of the PCCL and the typical CCL methods, we programmed the CPU-based CCL proposed by Hu et al. [34] and GPU-based multi-pass CCL methods proposed by Hawick et al. [38], which were implemented in the 768 $\times$ 576 resolution dataset. In these methods, the label array of a labeling execution was selected based on the propagating neighboring clique definition in Figure 8b. Figure 14 shows the relation between processing speeds and iteration times of these methods.

Because the CPU thread scanned the video images pixel by pixel during each iteration, many labeling iterations were required so that the labeling speed was slow when the foreground pixels occupied large regions in high-resolution images. As shown in Figure 14a, the speed of the CPU-based CCL decreased faster when more propagating iterations were executed. The average labeling speed performance of the CPU-based method was 41.31 fps with 14.40 iterations. Approximately, if more than 29 iterations were executed, the labeling speed would be reduced to below 20 fps, which does not satisfy the real-time requirement.

Different from CPU programming, the GPU-based labeling methods processed all pixels in parallel so that the labeling speeds did not decrease a lot even for the iteration increment. The GPU-based multi-pass CCL method could be implemented in real time for low-resolution images. However, this method required a large number of iterations for the propagation for high-resolution images, because a GPU thread only updated the center pixel in a clique with the minimum label. Thus, labeling convergence became slow due to the ineffective propagation process. As shown in Figure 14b, 121.98 iterations were executed using the GPU CCL in our experiment, and the labeling speed was reduced to 27.57 fps on average. In our CCL implementation, the local and neighboring labels were updated with the minimum label, instead of only one local label updated by the GPU CCL. Although the ranges of the label array in the GPU CCL and the CPU CCL were same, in the GPU CCL, the propagation bandwidth was smaller so that more iterations were required.

Also, in our PCCL implementation, a GPU thread labeled its corresponding local and neighbor pixels with the minimum label among its clique. In the PCCL and GPU CCL, the label array ranges were same. We updated five labels by a GPU thread, more than only one label updated by the GPU CCL, so that the labeling convergence was accelerated. This way, the iteration times were reduced and the labeling speed increased. As shown in Figure 14c, 64.32 iterations were executed using the PCCL, and the labeling speed was 71.35 fps on average. Although our proposed PCCL executed more interactions than the CPU CCL, the parallel computing mechanism accelerated the labeling speed. Thus, the PCCL was able to provide the best performance for high-resolution images.

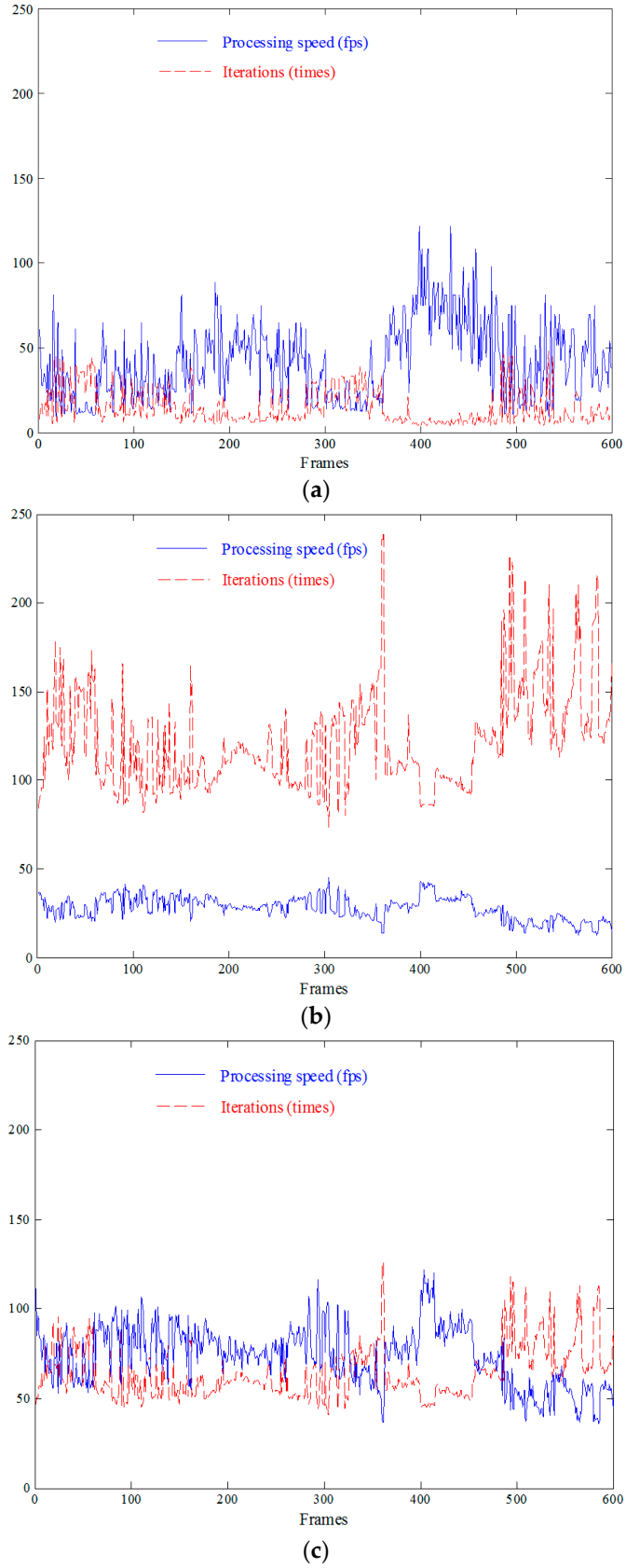


Figure 14. The relation between the labeling speed and the iteration times, executed in the dataset of 768 × 576 resolution. (**a**) The performance of the CPU-based CCL; (**b**) the performance of the GPU-based multi-pass CCL; (**c**) the performance of the proposed PCCL.

4.3. Sustainable Energy Management Application

Figure 15 illustrates a video surveillance application for a sustainable energy management system using the proposed GPU-accelerated foreground segmentation and labeling algorithms. In this project, a smart meter developed by Beijing Innolinks Technology Co. Ltd. (Beijing, China) [45], was utilized for ubiquitous controlling of the appliances, including air conditioning, water heater, and light.

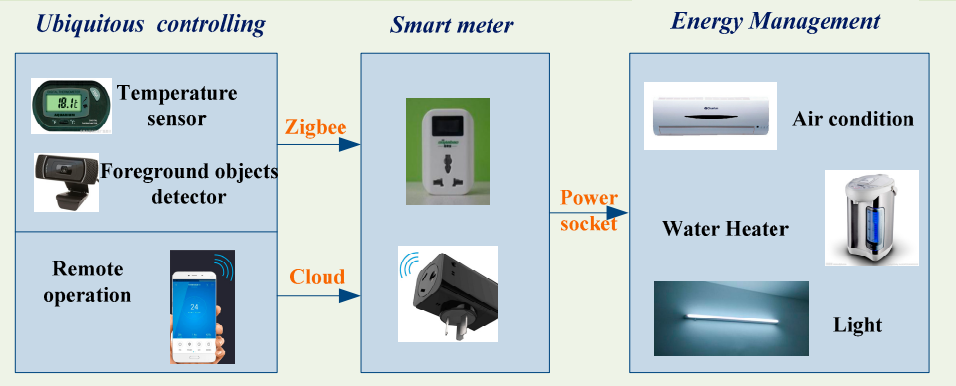


Figure 15. The proposed GPU-accelerated foreground segmentation and labeling algorithms applied in a sustainable energy management system with smart meters.

In the monitoring module, the environment information was recorded by multiple sensors and reported to the smart meter via the ZigBee communication. In our application, the ambient temperature was sensed by the thermometer mounted at the smart meter, and the foreground objects were detected by our proposed method. Based on these sensed datasets, the smart meter controlled the appliances for electricity saving by changing operation parameters or switching power supply. Meanwhile, the smart meter enabled the remote operation by smart phones through Cloud services for ubiquitous controlling.

In this project, our contribution was to develop a real-time foreground objects detection method for providing the smart meter with a decision-making basis. When the number of people in a workplace decreased, the smart meter turned up the temperature of the air condition. When a vehicle moved in a dark garage, the lightings around the car were turned on. When a person came into a room, the appliances connected with the smart meters were turned on. Using such smart energy management, the appliances were able to save around 20%–50% energy.

5. Conclusions

Recently, video surveillance systems, such as traffic analysis, indoor and outdoor security, people detection and tracking, and remote controlling, have been widely studied in intelligent monitoring as an important issue of future sustainable computing. This paper developed real-time foreground detection and labeling methods to segment the foreground pixels in video sequences. In the foreground detection process, we created a background model of pixel-wise color histograms in GPU memory to record color changes of all pixels. The background model was updating with the new registered frames, while the foreground pixels were detected if the pixels' color did not locate around the peaks of the corresponding histograms. To refine the segmentation results, we developed a PCCL algorithm to cluster the rough segmented foreground pixels into several distinguish connected blobs. A GPU thread labeled a foreground pixel and its neighboring foreground pixels with the minimum label among them. This propagation process was implemented iteratively until all label values were not changed. The noise blobs containing a small quantity of pixels were removed for increasing the foreground segmentation accuracy. The experimental results confirmed that the proposed method was able to deal with data-dependent situations in parallel and achieved real-time approach. The real-time and accurate foreground objects detection enhanced the auto-controlling efficiency of the smart meter for

realizing sustainable energy management. Using our proposed method, the connected components were clustered, but it was hard to distinguish two objects connecting with each other. In future, we will study an object tracking method for distinguishing the objects in the crossing situation.

Acknowledgments: This research was supported by Beijing Innolinks Technology Co. Ltd., by the National Natural Science Foundation of China (61503005), by Science and Technology Project of Beijing Municipal Education Commission (KM2015_10009006), by the young researcher foundation of NCUT (XN070027), by Advantageous Subject Cultivation Fund of NCUT, by SRF for ROCS, SEM.

Author Contributions: All the authors contributed equally to this work. All authors read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GPU	Graphics processing unit
CCL	Connected component labeling
PCCL	Parallel connected component labeling
CUDA	Compute Unified Device Architecture
FPGA	Field Programmable Gate Array
GMM	Gaussian mixture model
fps	Frames per second

References

1. Carroll, J.; Lyons, S.; Denny, E. Reducing household electricity demand through smart metering: The role of improved information about energy saving. *Energy Econ.* **2014**, *45*, 234–243. [CrossRef]
2. Saha, A.; Kuzlu, M.; Pipattanasomporn, M.; Rahman, S.; Elma, O.; Selamogullari, U.S.; Uzunoglu, M.; Yagcitekin, B. A Robust Building Energy Management Algorithm Validated in a Smart House Environment. *Intell. Ind. Syst.* **2015**, *1*, 163–174. [CrossRef]
3. Marinakis, V.; Karakosta, C.; Doukas, H.; Androulaki, S.; Psarras, J. A building automation and control tool for remote and real time monitoring of energy consumption. *Sustain. Cities Soc.* **2013**, *6*, 11–15. [CrossRef]
4. Ahn, H.; Jung, B.K.; Park, J.R. Effect of reagents on optical properties of asbestos and remote spectral sensing. *J. Converg.* **2014**, *5*, 15–18.
5. Erkan, B.; Nadia, K.; Adrian, F.C. Augmented reality applications for cultural heritage using Kinect. *Hum. Cent. Comput. Inf. Sci.* **2015**, *5*, 1–18.
6. Bayona, A.; SanMiguel, J.C.; Martinez, J.M. Comparative evaluation of stationary foreground object detection algorithms based on background subtraction techniques. In Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance, Genova, Italy, 2–4 September 2009; pp. 25–30.
7. Kim, K.; Chalidabhongse, T.H.; Harwood, D.; Davis, L. Real-time foreground–background segmentation using codebook model. *Real Time Imaging* **2005**, *11*, 172–185. [CrossRef]
8. Bouwmans, T.; Porikli, F.; Horferlin, B.; Vacavant, A. *Background Modeling and Foreground Detection for Video Surveillance*; CRC Press: Boca Raton, FL, USA; Taylor and Francis Group: Boca Raton, FL, USA, 2014.
9. Sobral, A.; Vacavant, A. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Comput. Vis. Image Underst.* **2014**, *122*, 4–21. [CrossRef]
10. Zhong, B.; Chen, Y.; Chen, Y.; Ji, R.; Chen, Y.; Chen, D.; Wang, H. Background subtraction driven seeds selection for moving objects segmentation and matting. *Neurocomputing* **2013**, *103*, 132–142. [CrossRef]
11. Das, S.; Konar, A. Automatic image pixel clustering with an improved differential evolution. *Appl. Soft Comput.* **2009**, *9*, 226–236. [CrossRef]
12. Song, W.; Cho, K. Real-time terrain reconstruction using 3D flag map for point clouds. *Multimed. Tools Appl.* **2015**, *74*, 3459–3475. [CrossRef]
13. Premanand, P.G.; Nilkanth, B.C. Content Based Dynamic Texture Analysis and Synthesis Based on SPIHT with GPU. *J. Inf. Process. Syst.* **2016**, *12*, 46–56.
14. Komura, Y.K.; Okabe, Y.T. GPU-based Swendsen–Wang multi-cluster algorithm for the simulation of two-dimensional classical spin systems. *Comput. Phys. Commun.* **2012**, *183*, 1155–1161. [CrossRef]

15. Song, W.; Wu, D.; Xi, Y.L.; Park, Y.W.; Cho, K. Motion-based skin region of interest detection with a real-time connected component labeling algorithm. *Multimed. Tools Appl.* **2016**. [CrossRef]
16. Cuevas, C.; Garcia, N. Efficient Moving Object Detection for Lightweight Applications on Smart Cameras. *IEEE Trans. Circuit Syst. Video Technol.* **2013**, *23*, 1–14. [CrossRef]
17. Ramasamy, T.; Asirvadam, V.; Sebastian, P. Detecting Background Setting For Dynamic Scene. In Proceedings of the IEEE 7th International Colloquium on Signal Processing and Its Applications, Penang, Malaysia, 4–6 March 2011; pp. 146–150.
18. Casares, M.; Velipasalar, S.; Pinto, A. Light-weight salient foreground detection for embedded smart cameras. *Comput. Vis. Image Underst.* **2010**, *114*, 1223–1237. [CrossRef]
19. Azmat, S.; Wills, L.; Wills, S. Temporal Multi-Modal Mean. In Proceedings of the 2012 IEEE Southwest Symposium on Image Analysis and Interpretation, Santa Fe, NM, USA, 22–24 April 2012; pp. 73–76.
20. Gallego, J.; Pardàs, M.; Haro, G. Enhanced foreground segmentation and tracking combining Bayesian background, shadow and foreground modeling. *Pattern Recognit. Lett.* **2012**, *33*, 1558–1568. [CrossRef]
21. Zamalieva, D.; Yilmaz, A. Background subtraction for the moving camera: A geometric approach. *Comput. Vis. Image Underst.* **2014**, *127*, 73–85. [CrossRef]
22. Jian, L.H.; Wang, C.; Liu, Y.; Liang, S.S.; Yi, W.D.; Shi, Y. Parallel data mining techniques on Graphics Processing Unit with Compute Unified Device Architecture(CUDA). *J. Supercomput.* **2013**, *64*, 942–967. [CrossRef]
23. Wilson, B.; Tavakkoli, A. An Efficient Non Parametric Background Modeling Technique with CUDA Heterogeneous Parallel Architecture. In Proceedings of the 11th International Symposium of International Symposium on Visual Computing, Las Vegas, NV, USA, 14–16 December 2015; pp. 210–220.
24. Stauffer, C.; Grimson, W.E.L. Adaptive background mixture models for real-time tracking. In Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, CO, USA, 23–25 June 1999; pp. 252–258.
25. Pham, V.; Vo, P.; Thanh, H.V.; Hoai, B.L. GPU Implementation of Extended Gaussian Mixture Model for Background Subtraction. In Proceedings of the IEEE International Conference on Computing and Telecommunication Technologies, Hanoi, Vietnam, 1–4 November 2010; pp. 1–4.
26. Boghdady, R.; Salama, C.; Wahba, A. GPU-accelerated real-time video background subtraction. In Proceedings of the 10th IEEE International Conference on Computer Engineering & Systems, Cairo, Egypt, 23–24 December 2015; pp. 34–39.
27. Fukui, S.; Iwahori, Y.; Woodham, R. GPU Based Extraction of Moving Objects without Shadows under Intensity Changes. In Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 4165–4172.
28. Griesser, A.; Roeck, S.D.; Neubeck, A. GPU-Based Foreground-Background Segmentation using an Extended Colinearity Criterion. In Proceedings of the 2005 Vision, Modeling, and Visualization, Erlangen, Germany, 16–18 November 2005; pp. 319–326.
29. Cheng, L.; Gong, M. Real Time Background Subtraction from Dynamics Scenes. In Proceedings of the IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 27 September–4 October 2009; pp. 2066–2073.
30. Wang, J.; Lu, Y.; Gu, L.; Zhoua, C.; Chai, X. Moving object recognition under simulated prosthetic vision using background-subtraction-based image processing strategies. *Inf. Sci.* **2014**, *277*, 512–524. [CrossRef]
31. Appiah, K.; Hunter, A.; Dickinson, P.; Meng, H. Accelerated hardware video object segmentation: From foreground detection to connected components labeling. *Comput. Vis. Image Underst.* **2010**, *114*, 1282–1291. [CrossRef]
32. Jiang, H.; Yu, T.; Tian, C.; Tan, G.; Wang, C. CONSEL: Connectivity-based Segmentation in Large-Scale 2D/3D Sensor Networks. In Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM 2012), Orlando, FL, USA, 25–30 March 2012; pp. 2086–2094.
33. He, L.F.; Chao, Y.Y.; Suzuki, K.J.; Wu, K.S. Fast connected-component labeling. *Pattern Recognit.* **2009**, *42*, 1977–1987. [CrossRef]
34. Hu, W.C.; Yang, C.Y.; Huang, D.Y. Robust real-time ship detection and tracking for visual surveillance of cage aquaculture. *J. Vis. Commun. Image Represent.* **2011**, *22*, 543–556. [CrossRef]
35. Hu, W.C.; Chen, C.H.; Chen, T.Y.; Huang, D.Y.; Wu, Z.C. Moving object detection and tracking from video captured by moving camera. *J. Vis. Commun. Image Represent.* **2015**, *30*, 164–180. [CrossRef]

36. Kalentev, O.; Rai, A.; Kemnitzb, S.; Schneider, R. Connected component labeling on a 2D grid using CUDA. *J. Parallel Distrib. Comput.* **2011**, *71*, 615–620. [CrossRef]
37. Hashmi, M.F.; Pal, R.; Saxena, R.; Keskar, A.G. A new approach for real time object detection and tracking on high resolution and multi-camera surveillance videos using GPU. *J. Cent. South Univ.* **2016**, *23*, 130–144. [CrossRef]
38. Hawick, K.A.; Leist, A.; Playne, D.P. Parallel graph component labelling with GPUs and CUDA. *Parallel Comput.* **2010**, *36*, 655–678. [CrossRef]
39. Cabaret, L.; Lacassagne, L.; Etiemble, D. Parallel Light Speed Labeling: An efficient connected component algorithm for labeling and analysis on multi-core processors. *J. Real Time Image Proc.* **2016**. [CrossRef]
40. Ferryman, J.; Shahrokni, A. PETS2009: Dataset and challenge. In Proceedings of the Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Snowbird, UT, USA, 7–9 December 2009; pp. 1–6.
41. Boragno, S.; Boghossian, B.; Black, J.; Makris, D.; Velastin, S. A DSP-based system for the detection of vehicles parked in prohibited areas. In Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, London, UK, 5–7 September 2007; pp. 260–265.
42. Zhong, J.; Sclaroff, S. Segmenting foreground objects from a dynamic textured background via a robust Kalman filter. In Proceedings of the Ninth IEEE International Conference on Computer Vision, Nice, France, 13–16 October 2003; pp. 44–50.
43. Toyama, K.; Krumm, J.; Brumitt, B.; Meyers, B. Wallflower: Principles and Practice of Background Maintenance. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; pp. 255–261.
44. Shah, M.; Deng, J.D.; Woodford, B.J. A Self-adaptive CodeBook (SACB) model for real-time background subtraction. *Image Vis. Comput.* **2014**, *38*, 52–64. [CrossRef]
45. Beijing Innolinks Technology Co. Ltd. Available online: http://www.innolinks.cn (accessed on 12 March 2014).