

Multimodal Mean Adaptive Backgrounding for Embedded Real-Time Video Surveillance

S. Apewokin, B. Valentine, L. Wills, S. Wills
Georgia Institute of Technology
Atlanta, GA, U.S.A.
scott.wills@ece.gatech.edu

A. Gentile
Dipartimento di Ingegneria Informatica
University of Palermo
Palermo, Italy

Abstract

Automated video surveillance applications require accurate separation of foreground and background image content. Cost sensitive embedded platforms place real-time performance and efficiency demands on techniques to accomplish this task. In this paper we evaluate pixel-level foreground extraction techniques for a low cost integrated surveillance system. We introduce a new adaptive technique, multimodal mean (MM), which balances accuracy, performance, and efficiency to meet embedded system requirements. Our evaluation compares several pixel-level foreground extraction techniques in terms of their computation and storage requirements, and functional accuracy for three representative video sequences. The proposed MM algorithm delivers comparable accuracy of the best alternative (Mixture of Gaussians) with a 6X improvement in execution time and an 18% reduction in required storage.

1. Introduction

Techniques for automated video surveillance utilize robust background modeling algorithms to identify salient foreground objects. Typically the current video frame is compared against a background model representing elements of the scene that are stationary or changing in uninteresting ways (such as rippling water or swaying branches). Foreground is determined by locating significant differences between the current frame and the background model.

The availability of low-cost, portable imagers and new embedded computing platforms makes video surveillance possible in new environments. However, situations in which a portable, embedded video surveillance system is most useful (e.g., monitoring outdoor and/or busy scenes) also pose the greatest challenges. Real-world scenes are characterized by changing illumination and shadows, multimodal features (such as rippling waves and rustling leaves), and frequent, multilevel occlusions. To extract foreground in these dynamic visual environments,

adaptive multimodal background models are frequently used that maintain historical scene information to improve accuracy. These methods are problematic in real-time embedded environments where limited computation and storage restrict the amount of historical data that can be processed and stored.

In this paper, we examine several representative pixel-based background modeling techniques in this real-time embedded environment. They are evaluated in terms of computational cost, storage, and extracted foreground accuracy. The techniques range from simple, computationally inexpensive methods, such as frame differencing and mean/median temporal filters [1], to more complex methods, including the multimodal Mixture of Gaussians (MoG) [2] approach. In this comparative evaluation, we include a new proposed approach, Multimodal Mean (MM), for real-time background modeling. Our technique achieves accuracy comparable to multimodal MoG techniques but with a significantly lower execution time. For our testbed, we employ commercial-of-the-shelf components to build a low-cost, low-power, and portable embedded platform. Our results demonstrate that our proposed MM algorithm achieves competitive real-time foreground accuracy under a variety of outdoor and indoor conditions with the limited computation and storage of a low-cost embedded platform.

2. Related Work

A variety of techniques exists for background subtraction; see [1], [3], and [4] for recent comprehensive surveys. *Frame differencing* compares pixels in the current video frame with corresponding pixels in the previous frame. If the difference between the pixels is above a given threshold, then that pixel is identified as foreground. While computationally inexpensive, this method is prone to the foreground aperture problem [5] and cannot handle dynamic background elements, such as swaying tree branches.

Sliding window-based (or *nonrecursive* [1]) techniques keep a record of the w most recent image frames. The background is represented as the mean or median of the frames in the buffer. Foreground is determined either by

determining if the current image pixel deviates by a fixed threshold away from the background model or, if it is within some standard deviation of the background. This type of technique is more memory intensive as it requires w image frames of storage per processed image.

Recursive techniques [1] utilize only the current frame and parametric information accumulated from previous frames to separate background and foreground objects. These techniques typically employ weighted means or approximated medians and require significantly less memory than the sliding window techniques. An *approximated median* is computed in [6]. The background is initialized by declaring the first image frame as the median. When a new video frame is acquired, the current image pixel values are compared with those of the approximated median pixel values. If a pixel value is above the corresponding median value, then that approximate median pixel value is incremented by one, otherwise it is decremented by one. It is assumed that the approximated median frame will eventually converge to the actual median after a given number of image frames are analyzed [6]. In [7] and [8], a *weighted mean* is used, which takes a percentage of the background pixel and a percentage of the current pixel to update the background model. This percentage is governed by a user-defined learning rate that affects how quickly objects are assimilated into the background model.

Issues can arise with the described techniques when there are moving background objects, rapidly changing lighting conditions, and gradual lighting changes. The Mixture of Gaussians (MoG) and Wallflower approaches are designed to better handle these situations by storing *multimodal representations* of backgrounds that contain dynamic scene elements, such as trees swaying in the wind or rippling waves. The MoG approach maintains multiple data values for each pixel coordinate. Each data value is modeled as a Gaussian probability density function (pdf) with an associated weight indicating how much background information it contains. With each new image frame, the current image pixel is compared against the pixel values for that location. A match is determined based on whether or not the current pixel falls within 2.5 standard deviations of any of the pixel distributions in the background model [2].

Wallflower [5] uses a three-tiered approach to model foreground and background. Pixel, region, and frame-level information are obtained and analyzed. At the pixel-level, a linear predictor is used to establish a baseline background model. At the region-level, frame differencing, connected component analysis and histogram backprojection are used to create foreground regions. Multiple background models are stored at the frame-level to handle a sharp environmental change such as a light being switched on or off.

These techniques have limitations in either foreground extraction accuracy or real-time performance when applied to busy or outdoor scenes in resource-constrained embedded computing systems. Frame differencing and recursive backgrounding methods do not handle dynamic backgrounds well. Sliding window methods require significant memory resources for accurate backgrounding. The MoG approach requires significant computational resources for sorting and computations of standard deviations, weights, and pdfs.

In this paper, we propose a new backgrounding technique that has the multimodal modeling capabilities of MoG but at significantly reduced storage and computational cost. A related approach [9] implements multimodal backgrounding on a single-chip FPGA using a collection of temporal lowpass filters instead of Gaussian pdfs. A similar background weight, match, and updating scheme as the MoG is maintained, with simplifications to limit the amount of floating-point calculations. In contrast to MoG and [9], we use a linear parameter updating scheme as opposed to nonlinear updates of weights and pixel values, and we make use of information about recency of background pixel matches. Updating the background model information in this manner allows for efficient storage of a pixel's long-term history.

3. Multimodal Mean Background Technique

We propose a new adaptive background modeling technique, called *multimodal mean*, which models each background pixel as a set of average possible pixel values. In background subtraction, each pixel I_t in the current frame is compared to each of the background pixel means to determine whether it is within a predefined threshold of one of them. Each pixel value is represented as a three-component color representation, such as an RGB or HSI vector. In the following, $I_{t,x}$ represents the x color component of a pixel in frame t (e.g., $I_{t,red}$ denotes the red component of I_t). The background model for a given pixel is a set of K mean pixel representations, called *cells*. Each cell contains three mean color component values. An image pixel I_t is a background pixel if each of its color components $I_{t,x}$ is within a predefined threshold for that color component E_x of one the background means.

In our embedded implementation, we chose $K = 4$ cells and use an RGB color representation. Each background cell B_i is represented as three running sums for each color component $S_{i,t,x}$ and a count $C_{i,t}$ of how many times a matching pixel value has been observed in t frames. At any given frame t , the mean color component value is then computed as $\mu_{i,t,x} = S_{i,t,x} / C_{i,t}$.

More precisely, I_t is a background pixel if a cell B_i can be found whose mean for each color component x matches within E_x the corresponding color component of I_t :

$$\left(\bigwedge_x |I_{t,x} - \mu_{i,t-1,x}| \leq E_x \right) \wedge (C_{i,t-1} > T_{FG})$$

where T_{FG} is a small threshold number of times a pixel value can be seen and still considered to be foreground. (In our experiments, $T_{FG} = 3$ and $E_x = 30$, for $x \in \{R, G, B\}$.)

When a pixel I_t matches a cell B_i , the background model is updated by adding each color component to the corresponding running sum $S_{i,t,x}$ and incrementing the count $C_{i,t}$. As the background gradually changes, for example, due to lighting variations), the running averages will adapt as well. In addition, to enable long-term adaptation of the background model, all cells are periodically *decimated* by halving both the sum and the count every d (the decimation rate) frames. To be precise, when I_t matches a cell B_i , the cell is updated as follows:

$$S_{i,t,x} = (S_{i,t-1,x} + I_{t,x}) / 2^b$$

$$C_{i,t} = (C_{i,t-1} + 1) / 2^b$$

where $b = 1$ if $t \bmod d = 0$, and $b = 0$, otherwise.

Decimation is used to decay long-lived background components so that they do not permanently dominate the model, allowing the background model to adapt to the appearance of newer stationary objects or newly revealed parts of the background. It also plays a secondary role in the embedded implementation in preventing counts from overflowing their limited storage. (In the experiments reported in this paper, the decimation rate $d = 400$, so decimation does not come into play in the test sequences. However, it is necessary for longer-term adaptation.)

When a pixel I_t does not match cells at that pixel position, it is declared to be foreground. In addition, a new background cell is created to allow new scene elements to be incorporated into the background. If there are already K background cells, a cell is selected to be replaced based on the cell's overall count $C_{i,t}$ and a recency count $R_{i,t}$ which measures how often the background cell's mean matched a pixel in a recent window of frames. A sliding window is approximated by maintaining a pair of counts $(r_{i,t}, s_{i,t})$ in each cell B_i . The first $r_{i,t}$, starts at 0, is incremented whenever B_i is matched, and is reset every w frames. The second $s_{i,t}$, simply holds the maximum value of $r_{i,t}$ computed in the previous window:

$$r_{i,t} = \begin{cases} 0, & \text{when } t \bmod w = 0 \\ r_{i,t-1} + 1, & \text{when } B_i \text{ matches } I_t \text{ and } t \bmod w \neq 0 \end{cases}$$

$$s_{i,t} = \begin{cases} r_{i,t-1}, & \text{when } t \bmod w = 0 \\ s_{i,t-1}, & \text{otherwise.} \end{cases}$$

Recency $R_{i,t} = r_{i,t} + s_{i,t}$ provides a measure of how often a pixel matching cell B_i was observed within a recent window. The $s_{i,t}$ component allows information to be

carried over across windows so that recency information is not completely lost at window transitions. When a new cell is created and added to a background model that already has K cells, the cell to be replaced is selected from the subset of cells seen least recently, i.e., cells whose recency $R_{i,t} < w/K$. From this set, the cell with the minimum overall count $C_{i,t}$ is selected for replacement. If all cells have a recency count $R_{i,t} > w/K$ (in the rare event that all cells are observed equally often over an entire window), then the cell with lowest $C_{i,t}$ is replaced. (In our experiments, we chose $w = 32$.)

4. Experiment

The backgrounding techniques are evaluated using three representative test sequences executing on an embedded execution platform. Each technique is compared in terms of image quality and accuracy (false positives and false negatives) as well as execution cost (execution time and storage required). The evaluated techniques include:

- frame differencing
- approximated median
- sliding window median
- weighted mean
- sliding window mean
- mixture of Gaussians (MoG)
- multimodal mean (MM)

The test suite includes two standard test sequences and a longer outdoor sequence captured using an inexpensive webcam (see Table 1). All sequences have a frame size of 160×120 .

Table 1: Test Sequences

Sequence	# Frames	Sampled Frame
Waving Tree	281	247
Bootstrapping	1000	299
Outdoors	201	190

The standard sequences, “Waving Tree” and “Bootstrapping,” are from the Wallflower benchmarks [5], using the same sampled frame and associated ground truth found in the published benchmarks. They contain difficult challenges for backgrounding algorithms. “Waving Tree” contains dynamic background in the form of a wind-blown tree with swaying branches and leaves. “Bootstrapping” lacks a “foreground-free” preamble for construction of the initial background model. This requires learning the background in the presence of continually changing foreground. These sequences are choreographed to present specific backgrounding problems. We also collected a longer sequence with dynamic background and the continuous presence of foreground objects. This sequence contains an outdoor scene with varying illumination, moving trees, and

subjects moving in varying patterns and positions. It was captured at 640×480 resolution at one frame per second. Afterward, the sequence was resized to 160×120, a sample frame was selected, and its ground truth was manually derived.

Table 2 lists the algorithm parameters used in the experiments. Experiment parameters and thresholds were held constant for all sequences. The MoG method incorporated $K=4$ Gaussians while the MM method utilized $K=4$ cells. The sliding window implementations use a buffer size of four for comparable memory requirements.

Table 2: Algorithm Parameters

Algorithm	Parameters
Mean/Median (SW)	$ \text{window} = 4$
Weighted Mean	$\alpha = 0.1$ for $u_t = (1-\alpha) * u_{t-1} + \alpha x_t$
Mixture of Gaussians (MoG)	$K=4$ modes, initial weight $w = 0.02$, learning rate $\alpha = 0.01$, weight threshold $T = 0.85$.
Multimodal Mean	$K=4, E_x = 30$ for $x \in \{R, G, B\}$, $T_{FG} = 3, d = 400, w = 32$

Our execution platform is an eBox-2300 Thin Client VESA PC [10] running Windows Embedded CE 6.0. The eBox, shown in Figure 1, incorporates a fanless Vortex86 SoC [11] (includes a 200MHz x86 processor that dissipates < 3 Watts) plus 128MB SDRAM (PC133), three USB ports, a 10/100 Ethernet port, and a compact flash slot. The platform is 11.5 × 11.5 × 3.5 cm in size, weighs 505g, and is designed for low power operation. Because of its limited 128MB internal memory, we constructed a customized lightweight kernel occupying approximately 19MB. Image sequences are also downloaded prior for each evaluation series.



Figure 1: eBox-2300 (from DMP Electronics Inc.)

Each backgrounding technique is implemented in C and compiled for Windows CE using Microsoft Studio. Algorithm data storage is limited to 40MB. This affects the variable window size for sliding window methods and

the number of modes for multimodal techniques.

5. Results and Evaluation

The accuracy and image quality of each method is compared in Figure 2 and Figure 3.

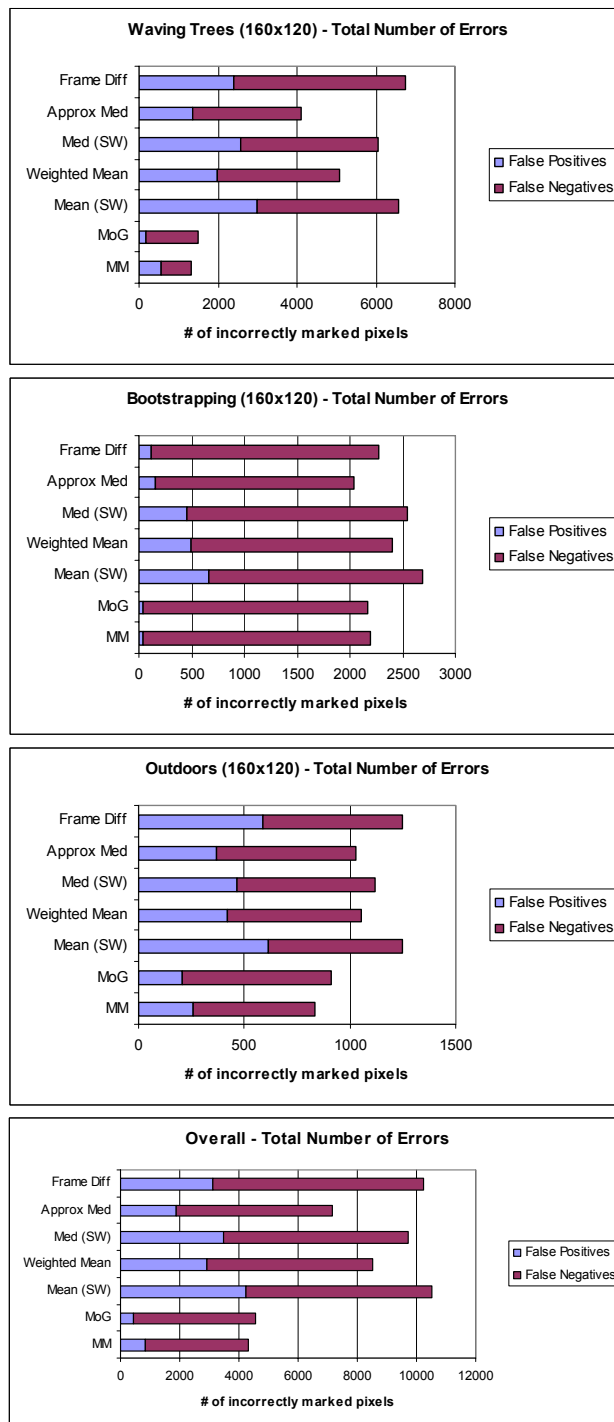


Figure 2: Backgrounding algorithm accuracy

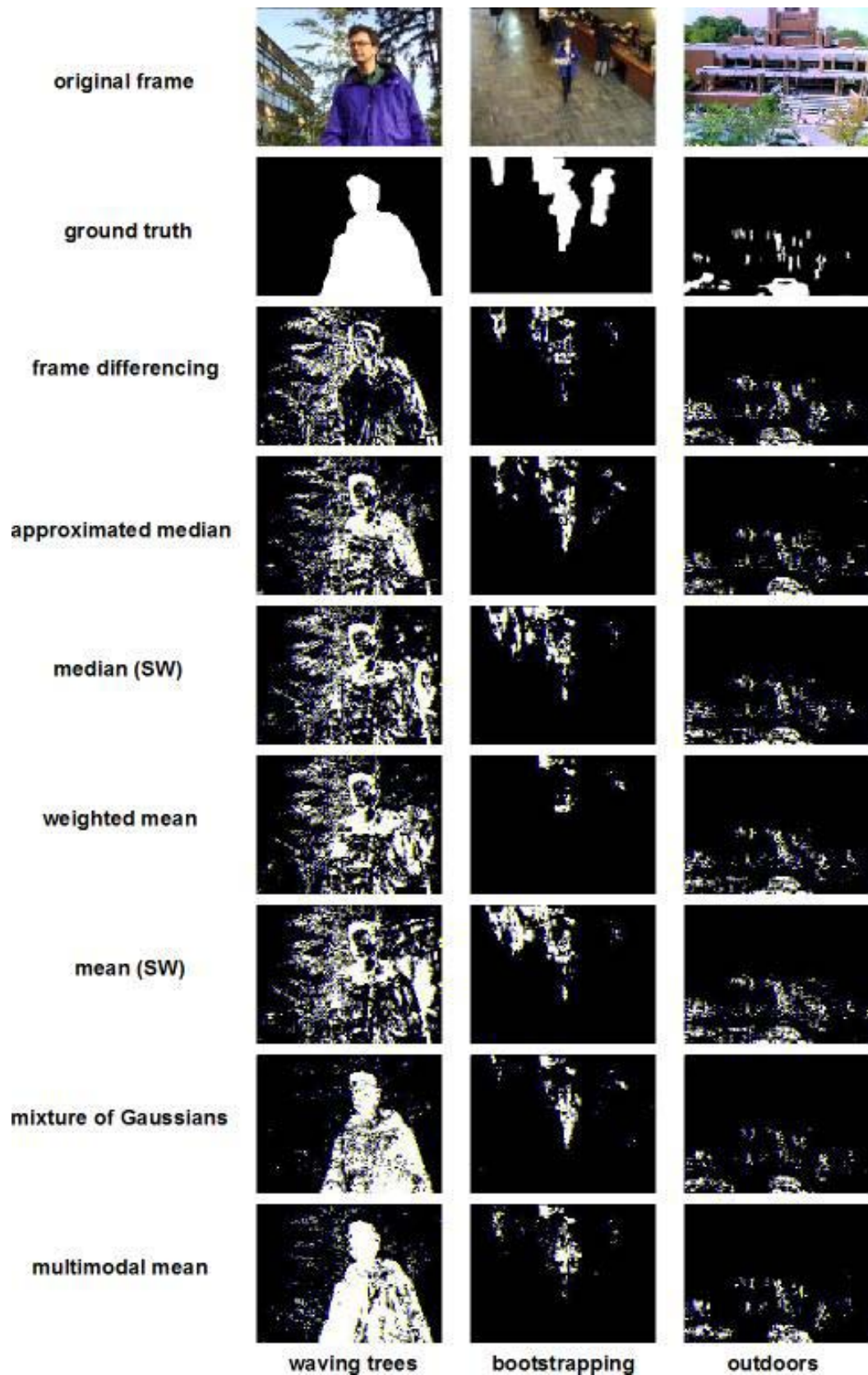


Figure 3: Image quality comparison of backgrounding techniques.

Figure 2 summarizes accuracy for each technique. False positives indicate foreground identified outside the highlighted (white) regions of the ground truth. False negatives result from background detected in ground truth identified foreground. While these counts do not provide a complete measure of foreground usefulness (e.g., often incomplete foreground can be “filled in”), lower numbers of false positives and negatives are usually desirable. Generally, MoG and MM demonstrate comparable accuracy that is superior to the other methods.

Figure 3 displays the image quality for each backgrounding technique. Multimodal methods (MoG and MM) generally exhibit the lowest number of errors across the sequences. False positives are significantly lower for the multimodal methods.

In “Waving Trees”, only the multimodal techniques incorporate the moving tree into the background. In “Bootstrapping”, all techniques are able to detect elements of the foreground identified in the ground truth. Unfortunately, the sliding window and weighted mean methods also identify reflected light on the floor (false positives). “Outdoors” features a large number of foreground elements as well as moving trees. Both multimodal techniques have significantly higher false positive accuracy.

Table 3 lists average processing times per frame, average frame rates, and storage requirements for each method executing on the test platform. Because the sequence frames originated from standard files rather than camera output, I/O requirements are not included in these figures.

Table 3: Algorithm performance on test platform.

Algorithm	Time (ms)	Rate (fps)	Storage (words/pixel)
Frame Differencing	7.6	132.0	1: packed RGB
Approximated Median	8.5	117.3	1: packed RGB
Median (SW)	69.2	14.4	3: 3 char × 4
Weighted Mean	26.8	37.3	1: packed RGB
Mean (SW)	28.2	35.5	3: 3 char × 4
MoG	273.6	3.7	22: 5 FP × 4 modes + 2 int
Multimodal Mean	43.9	22.8	18: (4 int + 2 char) × 4 cells

The results showed that our MM method executes 6.2× faster than the MoG technique, while providing comparable image quality and accuracy. It also requires 18% less storage per pixel and uses only integer operations. Although many of the other methods offered lower execution times and storage requirements, their accuracy is insufficient for many applications.

6. Conclusion

This paper compares several backgrounding techniques for time sensitive processing on embedded computing platforms. We have proposed a technique that combines the multimodal features of the mixture of Gaussians with simple pixel evaluation computations involving sums and averages. The Multimodal Mean method is able to achieve faster execution and lower storage requirements than mixture of Gaussians while providing comparable accuracy and output image quality.

References

- [1] Cheung, S. and Kamath, C. “Robust techniques for background subtraction in urban traffic video,” *Video Communications and Image Processing*, Volume 5308, pp. 881-892, SPIE Electronic Imaging, San Jose, January 2004.
- [2] C. Stauffer and W.E.L. Grimson, “Adaptive background mixture models for real-time tracking”, *Computer Vision and Pattern Recognition*, pp 246-252, June 1999.
- [3] Radke, R.J., Andra, S., Al-Kofahi, O., Roysam, B., “Image change detection algorithms: A systemic survey,” *IEEE Trans. on Image Processing*, 14(3) pp. 294-307, March 2005.
- [4] Piccardi, M., “Background subtraction techniques: a review,” *IEEE International Conference on Systems, Man and Cybernetics*, Vol 4., pp. 3099-3104, October 2004.
- [5] Toyama, K., Krumm, J., Brummitt, B., and Meyers, B., “Wallflower: Principles and Practices of Background Maintenance,” in *Proc. of ICCV (1)*, pp. 255-261, 1999; Wallflower benchmarks available online at research.microsoft.com/~jckrumm/WallFlower/TestImages.htm.
- [6] N. McFarlane and C.Schofield, “Segmentation and tracking of piglets in images,” *Machine Vision and Applications* 8(3), pp. 187-193, 1995.
- [7] Jabri, S., Duric, Z., Wechsler, H., and Rosenfeld, A., “Detection and location of people in video images using adaptive fusion of color and edge information,” *IEEE International Conference on Pattern Recognition*, pp. 627-630, vol 4., September 2000.
- [8] Wren, C. R., Azarbayejani, A., Darell, T., Pentland, A.P., “Pfinder: real-time tracking of human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), pp.780-785, July 1997.
- [9] Appiah, K., Hunter, A., “A single-chip FPGA implementation of real-time adaptive background model,” *IEEE International Conference on Field-Programmable Technology*, pp. 95-102, December 2005.
- [10] DMP Electronics Inc., “VESA PC eBox-2300 Users Manual,” September 2006.
- [11] Silicon Integrated Systems Corp., “SiS55x Family Datasheet,” Rev 0.9, 14 March 2002.