Kalman Filter Based Video Background Estimation

Jesse Scott
jesse.scott@heatinc.com
Electrical and Computer Engineer
Digital Imaging Group
www.heatinc.com
heatfusion
10416 Theodore Green Boulevard
White Plains, Maryland 20695

Michael A. Pusateri, Ph. D.
mpusateri@engr.psu.edu
Director, Senior Research Associate
Integrated Design Services
www.ecs.psu.edu/design/
Penn State University
149 Hammond Building
University Park, Pennsylvania 16802

Duane Cornish
dcc194@psu.edu
Ph.D. Candidate
Multidimensional Image Processing Lab
http://www.mipl.ee.psu.edu/
Penn State University
304 Electrical Engineering West
University Park, Pennsylvania 16802

***Abstract*- Transferring responsibility for object tracking in a video scene to computer vision rather than human operators has the appeal that the computer will remain vigilant under all circumstances while operator attention can wane. However, when operating at their peak performance, human operators often outperform computer vision because of their ability to adapt to changes in the scene.**

While many tracking algorithms are available, background subtraction, where a background image is subtracted from the current frame to isolate the foreground objects in a scene, remains a well proven and popular technique. Under some circumstances, a background image can be obtained manually when no foreground objects are present. In the case of persistent surveillance outdoors, the background has a time evolution due to diurnal changes, weather, and seasonal changes. Such changes render a fixed background scene inadequate.

We present a method for estimating the background of a scene utilizing a Kalman filter approach. Our method applies a one-dimensional Kalman filter to each pixel of the camera array to track the pixel intensity. We designed the algorithm to track the background intensity of a scene assuming that the camera view is relatively stationary and that the time evolution of the background occurs much slower than the time evolution of relevant foreground events. This allows the background subtraction algorithm to adapt automatically to changes in the scene.

The algorithm is a two step process of mean intensity update and standard deviation update. These updates are derived from standard Kalman filter equations. Our algorithm also allows objects to transition between the background and foreground as appropriate by modeling the input standard deviation. For example, a car entering a parking lot surveillance camera field of view would initially be included in the foreground. However, once parked, it will eventually transition to the background.

We present results validating our algorithm's ability to estimate backgrounds in a variety of scenes. We demonstrate the application of our method to track objects using simple frame detection with no temporal coherency.

I. Introduction

In this paper, we investigate a method for background estimation as a preface to background subtraction. Background subtraction is a basic operation in image and video processing and is the initial operation in many object tracking algorithms. For background subtraction to be automated, we must answer the question: "How do we get a background image without operator intervention?"

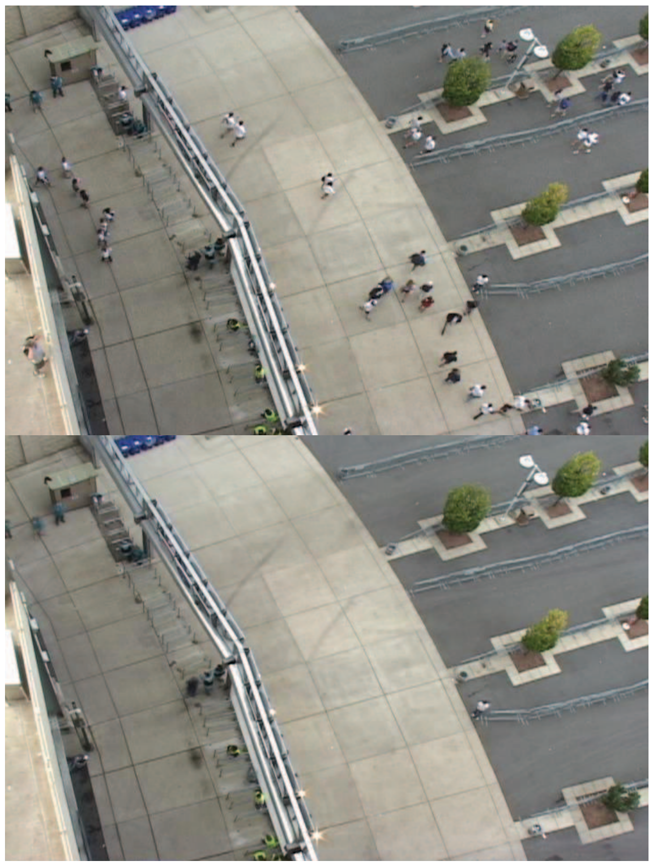
Figure 1. Kalman filter input and output results.

We show that one way to obtain an estimate of the background is by applying a matrix of one-dimensional estimation filters based on a derivation of the fundamental prediction-correction Kalman filter equations [1][2]. The algorithm estimates the background image of a particular scene without prior knowledge of the background or scene. The underlying assumption we make is simply that pixels in the background have intensities that do not evolve quickly in time, while foreground objects do. The timescale of slow versus fast evolution is controllable by adjusting the filter's time constant. Pixels with high intensity variances are segmented to the foreground, while pixels with low variances are segmented to the background. This process allows for operator free functionality while adapting to slow temporal variability.

A. Goals

The primary goals of this paper are to explain a simple background estimation method, demonstrate the use of this method for background subtraction during target detection, and to evaluate this technique for real-time hardware implementation. A secondary goal is to demonstrate and evaluate the algorithm's ability to adapt to changing imagery and scene content. All of these goals will be evaluated from the point of view of persistent surveillance using visual analysis alone.

B. Design Constraints

To accomplish the outlined goals, the current state-of-the-art methods were researched. The most promising of those methods were considered for their viability within the engineering constraints and a candidate method was prototyped in MATLAB. The design constraints are simple, but strict for this design. The constraints are as follows:

- The design must minimize resource usage and energy consumption.
- The design must utilize as little external memory resources as possible.
- The design must have intra-frame latency, preferably less than 100 pixels of latency.
- The design must provide an estimated background without user interaction.
- The design must work with stationary surveillance imagers with expandability to systems with ego-motion.

II. Background

A. Background Subtraction

The standard method for acquiring a background image is operator based manual capture of imagery when the operator believes there is only background content. This method has three downsides. The first is the need for operator intervention requiring active user observation. The second issue is the assumption that the operator can distinguish between foreground and background. The final downside is that the background, even if it is captured correctly, evolves over time as the result of environmental and temporal affects, causing a need for regular recapture of the background image.

Even with significant operator training, the background may have considerable errors with proper operator intervention. From experiential observation, a captured background is often stuck with unwanted foreground objects because a foreground free scene never occurs, resulting in an unclear separation between background and foreground.

B. Background Estimation

The background can be estimated to help alleviate some of the issues that arise from the standard method [3][4][5]. The simplest approach is a pixel by pixel intensity statistical estimation scheme. A simple version of this structure is a windowed mean utilizing N previous frames of history. Although very simple and straight forward, windowing has a temporal history only as long as the window depth causing issues related to the large amount of external memory resources required to retain and retrieve the entire history window. However, the algorithm will adapt to a temporally changing background and does not require user intervention, but still has an imperfect separation between foreground and background.

C. Kalman Filter Background Estimation

When evaluating Kalman based background estimation methods, the literature review revealed four primary categories that are associated with Kalman filters [3][4][6]. The four primary categories of background estimation methodologies are time evolution, local structure, local time evolution, and feedback time evolution. All are focused on a goal of generating a background image given imagery that constantly has foreground content. The literature review also presented three key techniques that define the primary categories of Kalman background estimation. The three techniques are temporal pixel-wise, spatial local-region, and state feedback processing. The category selected is the time evolution method that uses temporal pixel-wise processing. The choice is based on the uniform implementation as well as the simplicity of implementation. The choice meets all of the engineering requirements and minimizes resource usage as compared to the other categories.

D. Kalman Filter Background Estimation

The equations used are derived from the fundamental prediction/correction Kalman filter equations [1][2] in (1) through (5). The five Kalman filter equations are the state prediction in (1), error prediction in (2), gain in (3), state correction in (4), and error correction in (5). In these equations the following variables need defined:

- A matrix relates the state at the previous time step to the state at the current step.
- B matrix relates the optional control input to the state.
- Q represents the process noise covariance matrix.
- H matrix relates the state to the measurement.

- R represents the measurement noise covariance matrix.
- x is the state variable with k being the current and k-1 being the prior.
- u is the control variable with k being the current.
- z is the measurement with k being the current

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{1}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2}$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{4}$$

$$P_k = (I - K_k H)P_k^- \tag{5}$$

III. Approach

Our algorithm uses a one-dimensional Kalman filter at each pixel index that tracks the pixel intensity at each index. The Kalman filter update equation requires retention of the standard deviation prior and the mean prior for each pixel in the image [1]. The benefits of this algorithm include independence from scene content, long term and short term historical behavior, deterministic execution, and resource requirements independent of historical complexity. The standard deviation and prior intensity values are updated using Kalman filter prediction/correction equations manipulated to mean and standard deviation update equations.

A. *Selected Algorithm*

Figure 2 shows the overall flow of our object detection algorithm for automatically processing streaming or prerecorded video. Each step in the flow of the algorithm highlights the key algorithmic steps required to not only perform background estimation and subtraction but also group and indicate foreground objects in the image.

The initialization step sets the values for all user controlled variables and the prior mean and standard deviation of the Kalman filter update equations. Currently, the Kalman filter matrix priors are set to values that cause the first measurement to be used as a major portion of the next iterations estimate. It should be also noted that the variables for noise modeling are currently constant and uniform over the entire image. Adapting these variables to be a nonuniform matrix that fluctuates based on image statistics would improve the behavior of the background estimate as it adapts to changing conditions.

Once the new image is acquired, the update equations are executed to generate a new mean and standard deviation estimate. The mean estimate is then subtracted from the measurement to find the foreground objects in the scene. The difference image is not binary, but nearly emulates binary behavior. As part of the Kalman filter equations, a variable is included to accommodate the measurement noise. Although sensor noise is not intended to be included in this noise variable, it tends to naturally occur, resulting in a mean estimate that does not include sensor noise. The lack of noise in the estimate causes the difference image to contain the noise that existed in the measurement values.

The next step is to rectify the three color difference image into a single binary mask that identifies foreground and background only. This is accomplished via a three color to grayscale difference image followed by a simple threshold designed to accommodate the sensor noise. The binary image is then processed by marking the separate foreground groups by using a simple nearest neighbor association technique.

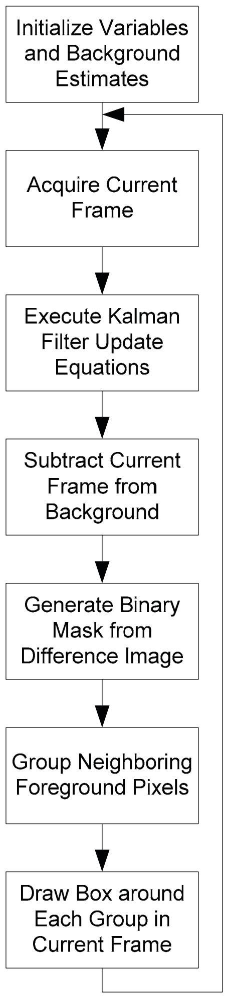
Initialize Variables and Background Estimates
Acquire Current Frame
Execute Kalman Filter Update Equations
Subtract Current Frame from Background
Generate Binary Mask from Difference Image
Group Neighboring Foreground Pixels
Draw Box around Each Group in Current Frame

Figure 2. Algorithmic Flow of Object Detection Algorithm.

The final step is to translate the groups based on the binary mask by indicating each group on the current frame. This is done by placing a bounding box around contiguous groups based on the grouping's width, height, and position. These boxes are overlaid to generate a simple indication of foreground objects. It should be observed that there is no temporal tracking of foreground objects, but frame by frame detection only. The overlay is intended to present visual indication of the quality of the background estimation as a tool for more complex computer vision and image processing techniques.

B. Derived Update Equations

The update equations (6) and (7) are derived from the fundamental versions presented in (1) through (5). The mean update (6) and standard deviation update (7) equations utilize variables that control certain aspects of the original equations. The following is a list of relations between variable matrices in the fundamental equations to variables in the derived update equations.

- The Q matrix is related to the d variable that is the process noise gain.
- The R matrix is related to the s variable that is the measurement noise gain.
- The variable g is the ratio of (s^2/d^2) used as a ratio-metric noise gain

$$\mu_{ij}[k] = \frac{(g_{ij} * \mu_{ij}[k-1]) + \left(\frac{z_{ij}[k]}{d_{ij}} * \sigma_{ij}[k-1]\right)}{(g_{ij} + \sigma_{ij}[k-1])} \tag{6}$$

$$\sigma_{ij}[k] = \frac{(g_{ij} * \sigma_{ij}[k-1])}{(g_{ij} + \sigma_{ij}[k-1])} \tag{7}$$

```
mu_k_Num =    ((((s.*s)./(d.*d)) .* mu_k) +
              ((sigma_k) .* (currentImage./d)));
mu_k_Denom =  (((s.*s)./(d.*d)) + (sigma_k));
mu_k =        mu_k_Num ./ mu_k_Denom;
```

Figure 3. Code for Kalman filter matrix mean update equation.

```
sigma_k =     ((s.*s)./(d.*d) .* (sigma_k)) ./
              ((s.*s)./(d.*d) + (sigma_k));
```

Figure 4. Code for Kalman filter matrix standard deviation update equation.

```
gray_inv_bg =  max(abs(curImage - mu_k), [], 3);
bin_fg_mask =  imclose(im2bw(gray_inv_bg, .25),
               strel('disk',2));
```

Figure 5. Code for foreground mask generation.

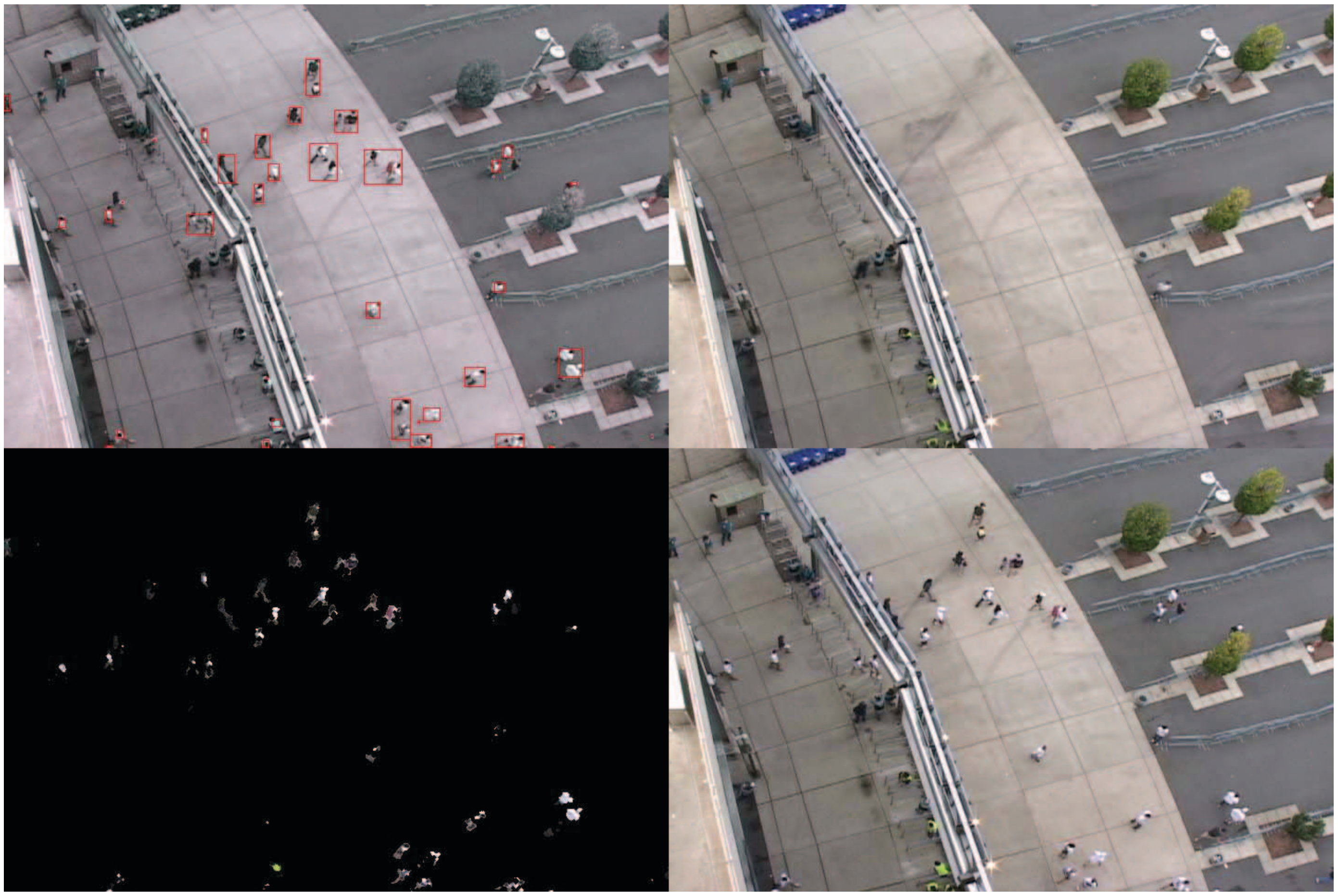

Figure 6. Frame from Beaver Stadium video sequence.

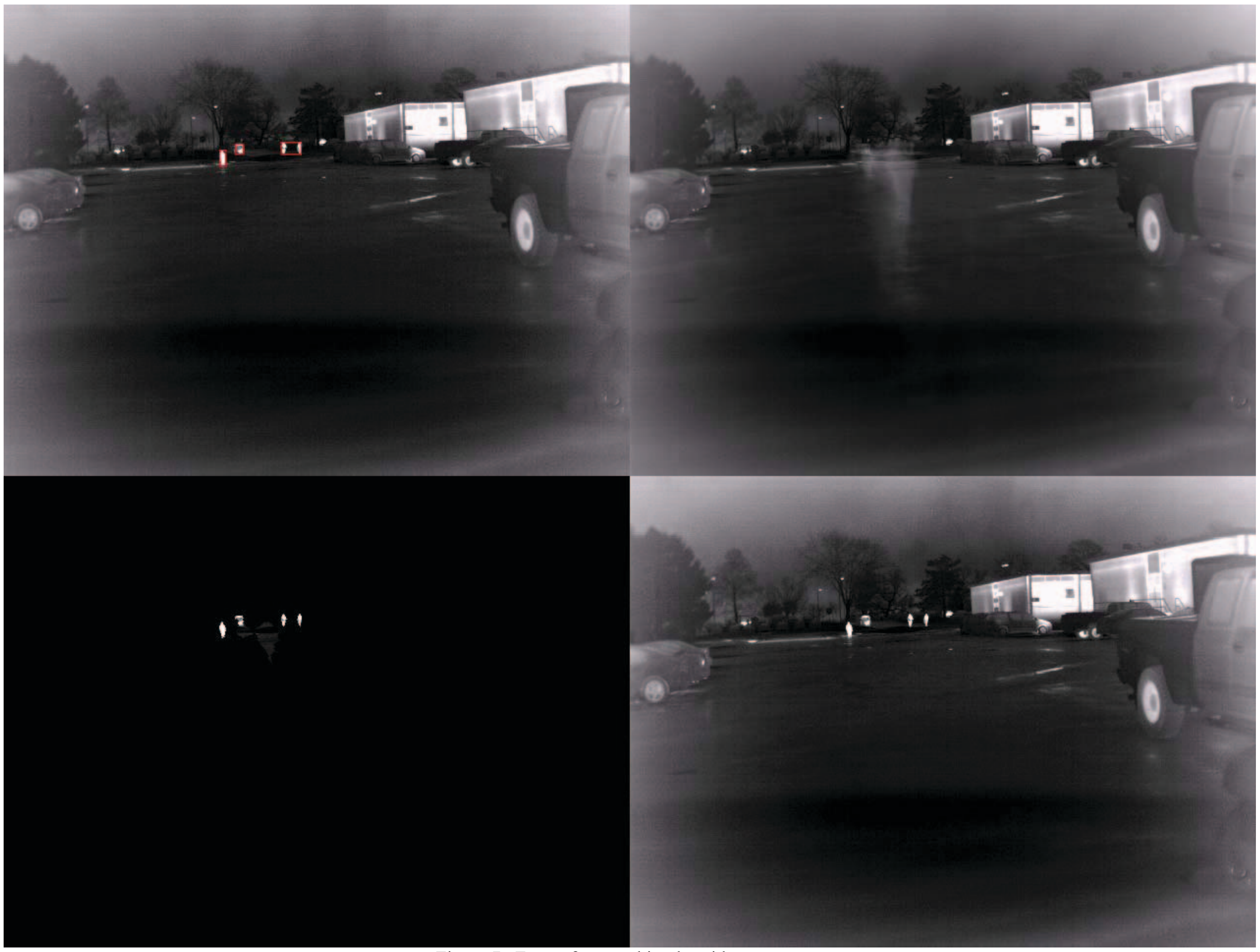
Figure 7. Frame from parking lot video sequence.

C. Implementation

The code depicted in Figures 3-5 was tested within the MATLAB environment. Support code was included for output formatting to video and imagery as well as marking and indicating the foreground objects in the image. This detection is simple and is not the focus of this research, but is necessary for properly indicating foreground objects.

IV. Results

The following results are select frames that present the results of processing two different video sequences. Additionally, the following examples show that this technique is functionally independent of resolution and wavelength. For all of the following examples, the imagery is organized into quads to present input and output as well as two intermediate steps of the algorithm. The organization of the quads is as follows:

- Top Left – Rectangle indicators over original
- Top Right – Estimated Background
- Bottom Left – Foreground of Original Only
- Bottom Right – Original Image

A. Stadium Entrance

The image shown in Figure 6 depicts a downward looking scene over Gate A at Beaver Stadium at Penn State. The camera used is a 3CCD visible band color sensor with a 720x480 resolution and 25 frames per second sampling. The daytime video sequence is about 40 seconds long where this frame is about 15 seconds into the sequence. The frame shown is meaningful because it depicts both the pros and cons of this estimation method. The patrons are clearly separated from the background but many of security/staff members at the gate are part of the background because they are stationary.

B. Parking Lot

This is a forward looking scene collected in a parking lot directed towards a crossing road. The camera used is a long-wave infrared sensor with a 640x480 resolution and 30 frames per second sampling. The video was captured after sunset during moderate rain showers and is about 150 seconds long. The frame shown is about 70 seconds into the scene and contains three people and one vehicle. The background has some minor artifacts, but they do not affect the detection process.

V. Analysis

A. *Performance*

The performance of the background estimation appears to be more than sufficient across multiple sequences. For lack of a ground truth baseline, no analytical or objective analyses were performed. Subjective analysis shows accuracy and consistency in the object detection as a direct result of the estimate of the background. The algorithm also appears to require no user configuration or manipulation even between different video sequence formats.

B. *Resources*

Because this design must be amendable to a real-time HDL based implementation, an initial resource analysis was performed for each of the top three resource intensive operations. The resources are defined by the following four categories: External RAM Size, External RAM bandwidth, Internal Logic, and Internal RAM. Since much of the resources are associated with the resolution of the input image, we will define the image as M pixels in width, N pixels in height, P pixel layers, and F frames per second. Since hardware resource definitions vary across platforms, the resources will be defined by the number of fundamental mathematical operations performed per pixel.

Kalman filter mean update equation requires a MxNxP array of doubles in external RAM space with a MxNxPxF bandwidth. The logic requirements to perform the calculation are two divisions, two multiplications, and two additions. The simple logic diagram for this operation is shown in Figure 8, where green blocks are incoming measurements, gray blocks are constants, and red blocks are variables. The latency of the system is clearly short, but depends on the pipelining implemented in the multiplication and division operations. Clearly, the three primary operations in serial order will result in total latency on the order of 10's of pixels.

The Kalman filter standard deviation update equation requires a MxNxP array of doubles in external RAM space with a MxNxPxF bandwidth. The logic requirements to perform the calculation are one division, one multiplication, and one addition. The simple logic diagram for this operation is shown in Figure 9, where gray blocks are constants and red blocks are variables.

Object detection equation requires a MxN array of bits in external RAM space with a MxNxF bandwidth. The logic requirements to perform the calculation are one max, one comparison, one subtraction, and one multiplexer. The simple logic diagram for this operation is shown in Figure 10, where green blocks are incoming measurements, gray blocks are constants, and red blocks are variables.

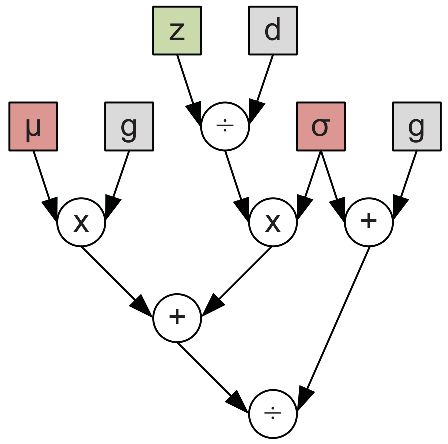

Figure 8. Logic diagram for mean update equation.

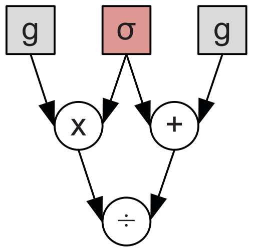

Figure 9. Logic diagram for standard deviation update equation.

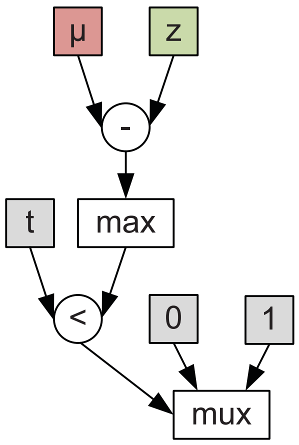

Figure 10. Logic diagram for the object detection equation.

C. *Future Improvements*

The current application can be expanded to allow ego-motion with simple azimuth information mapped to an array for all 360 degrees of rotation. This will increase the external resource requirements based on the field of view (FoV) and resolution of the sensor. In our examples, 40 degree FoV repeated over the 360 rotation will nine times increase the memory requirements. The bandwidth requirements will not increase; only the storage size. What would need investigation is the effect of nonuniform update intervals on (6) and (7). As the imager is panned only the region currently observed is uniform, the FoV that is added or removed from the current region will have a new sample, but the time interval will not be the same as the rest of the region.

Currently, our method processes imagery, but does not fully utilize all information available for controlling the algorithm. As noted earlier, the variables for noise modeling are constant and uniform over the entire image. Adapting these variables to be a nonuniform matrix that varies based on image statistics would improve the behavior of the background estimate as it adapts to changing conditions. To accomplish this change, more external and internal resources will be required.

To speed up the current implementation done in MATLAB, migration to an FPGA platform should provide all the resources required to perform this method on all the current state-of-the-art sensors resolutions, pixel depths, and frame rates. Although the examples were all prerecorded video, this algorithm is capable of processing a webcam quality video feed at 15 frames per second in the MATLAB environment on a 2.0 GHz Intel Core2 Duo laptop with 2GB of memory. Considering this fact, a hardware based implementation should have no technical hurdles at a data rate of 50x, far exceeding even current sensor technology data rates.

Currently, the foreground objects are simply detected from frame to frame with absolutely no temporal tracking. A simple improvement to the object detection would be to incorporate a simple tracker for the detected objects. This will not improve the fundamental behavior of background estimation, but will improve the overall object detection performance.

VI. Summary

Kalman background estimation provides sufficiently useful background information without user intervention. The equations are simple, straight forward, and adapt to widely varying conditions while requiring little to no configuration. This design provides an effective method for foreground extraction dramatically increasing the efficiency of the overall goal of computer vision. Overall, we feel that our Kalman filter based Background estimation performed quite well for an initial simplified prototype.

References

[1] G. Welch, G. Bishop, “An Introduction to the Kalman Filter,” IEEE Special Interest Group on Graphics (SIGGRAPH ‘01), Course 8, 2001.

[2] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” Transactions of the ASME–Journal of Basic Engineering, 82 (Series D), pages 35-45, 1960.

[3] S. Cheung, C. Kamath, “Robust techniques for background subtraction in urban traffic video,” Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 5308, pages 881-892, Jan., 2004.

[4] D. Koller, J. Weber, and J. Malik, “Robust multiple car tracking with occlusion reasoning,” IEEE European Conference on Computer Vision (ECCV'94), pages 189–196, Stockholm, Sweden, May 1994.

[5] Tsung-Han Tsai, Chung-Yuan Lin, De-Zhang Peng, Giua-Hua Chen, "Design and Integration for Background Subtraction and Foreground Tracking Algorithm," International Conference on Information Assurance and Security, pages 181-184, 2009.

[6] J. Zhong, S. Sclaroff, "Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter," IEEE International Conference on Computer Vision (ICCV'03), page 44, Volume 1, 2003.