

# Real-time Video Segmentation with VGA Resolution and Memory Bandwidth Reduction

Hongtu Jiang and Viktor Öwall  
Department of Electrosience  
CCCD, Lund University  
Lund, 22100, Sweden  
{htj,vikt}@es.lth.se

Håkan Ardö  
Department of Mathematics  
Lund University  
Lund, 22100, Sweden  
ardo@maths.lth.se

## Abstract

*This paper presents the implementation of a video segmentation unit used for embedded automated video surveillance systems. Various aspects of the underlying segmentation algorithm are explored and modifications are made with potential improvements of segmentation results and hardware efficiency. In addition, to achieve real-time performance with high resolution video streams, a dedicated hardware architecture with streamlined dataflow and memory access reduction schemes are developed. The whole system is implemented on a Xilinx FPGA platform, capable of real-time segmentation with VGA resolution at 25 frames per second. Substantial memory bandwidth reduction of more than 70% is achieved by utilizing pixel locality as well as wordlength reduction. The hardware platform is intended as a real-time testbench for observations of long term effects with different parameter settings, which is hard to achieve on a PC platform.*

## 1. Introduction

Automated video surveillance system is gaining substantial interests in the research community in recent years. This is partially due to the progress in technology scaling that enables more robust yet computationally intensive algorithms to be realized in reasonable performance. The advantage of surveillance automation over traditional TV based system lies in the fact that it is a self contained system capable of automatic information extraction, e.g. moving objects extraction and tracking. The result is a fully or semi automated surveillance systems, potentially cutting the cost of human resources observing the output from cameras. Typical applications may include both civilian and military scenarios, e.g. traffic control, security surveillance in banks or antiterrorism.

Crucial to all these applications is the quality of the video segmentation, which is a process of extracting objects of interest (foreground) from an irrelevant background scene. The foreground information, usually composed of moving objects, is passed on to later analysis units, where objects are tracked and their activities are analyzed. A wide range of segmentation algorithms have been proposed in the literature [2, 13, 9], with aimed robustness to different situations. A comparison is made in [12] on the segmentation qualities derived from various approaches. In fact, no perfect system exists to handle all kinds of issues within different background models. Furthermore, for realistic implementation of such system, trade-offs have to be made between system robustness (quality) and system performance (frame rate, resolution, etc.). In [11], a background model based on pixel wise multi-modal Gaussian distribution is proposed with the robustness to multi-modal background situations, which are quite common in both indoor and outdoor environments. A multi-modal background is caused by repetitive background object motion, e.g. swaying trees, flickering of the monitor etc. As a pixel lying in the region where repetitive motion occurs will generally consists of two or more background colors, the RGB value of that specific pixel changes over time. This will result in false foreground detection in most other approaches. Various modifications to the algorithm for potential improvements are also reported [5, 8, 3, 14, 15]. However, none of these works address the issue of the algorithm performance in terms of meeting real-time requirements with reasonable resolution. Pixel wise image processing is costly in computation and storage, let alone each pixel is characterized by several Gaussian distributions, each of which contains several parameters. In [11], only a frame rate of 11-13 FPS is obtained even for a small frame size of  $160 \times 120$  on an SGI O2 workstation. In our PC implementation on an AMD 4400+ processor, a frame rate of 4-6 FPS is observed for video sequences with  $352 \times 288$  resolution. In addition to

performance issues, we have found no studies on possible algorithm modifications that could lead to potentially better hardware efficiency.

In this paper, we present a dedicated hardware architecture capable of real-time segmentation with VGA resolution at 25 FPS. A variety of memory access reduction schemes are implemented, resulting in more than 70% memory bandwidth reduction. Furthermore, various modifications to the algorithm are made, with potential improvements of hardware efficiency. The paper is organized as follows. Section 2 and 3 discuss the original algorithm and possible modifications for hardware efficiency. The hardware architecture is presented in Section 4, together with the memory bandwidth reduction scheme explained in detail. Finally, the results and conclusions are covered in Section 5 and 6.

## 2 Gauss Mixture Background Model

The algorithm is briefly formulated as follows: Measured from consecutive video frames, the values of any pixel can be regarded as a Gaussian distribution. Characterized by mean and variance values, the distribution represents a location centered at its mean values in the RGB color space, where the pixel value is most likely to be observed over frames. A pixel containing several background object colors, e.g. the leaves of a swaying tree and a road, can be modeled with a mixture of Gaussian distributions with different weights. The weight of each distribution indicates the probability of matching a new incoming pixel. A match is defined as the incoming pixel within  $J$  times standard deviation off the center, where  $J$  is selected as 2.5 [11]. The higher the weight, the more likely the distribution belongs to the background. Mathematically, the portion of the Gaussian distributions belonging to the background is determined by

$$B = \underset{b}{\operatorname{argmin}} \left( \sum_{k=1}^b \omega_k > H \right), \quad (1)$$

where  $H$  is a predefined parameter and  $\omega_k$  is the weight of distribution  $K$ . If a match is found, the matched distribution is updated as:

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha \quad (2)$$

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t \quad (3)$$

$$\sigma^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t); \quad (4)$$

where  $\mu, \sigma$  are the mean and variance respectively,  $\alpha, \rho$  are the learning factors, and  $X_t$  are the incoming RGB values. The mean, variance and weight factors are updated frame

by frame. For those unmatched, the weight is updated according to

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1}, \quad (5)$$

while the mean and the variance remain the same. If none of the distributions are matched, the one with the lowest weight is replaced by a distribution with the incoming pixel value as its mean, a low weight and a large variance.

## 3 Algorithm Modifications

The algorithm works efficiently only in controlled environment. Many issues regarding algorithm weaknesses in different situations are addressed in many publications[5, 8, 3, 14]. In this section, instead of mainly focusing on improving algorithm robustness, we propose several modifications to the algorithm, with major concern on their impacts that could lead to potentially improved hardware efficiency. The modifications being made are covered in detail in the following sections and can be characterized into 3 categories: purely technical (Section 4.1, 4.2), related to the approach (Section 3.1, 3.2) and application specific (Section 3.3).

### 3.1 Color Space Transformation

In theory, multi-modal situations only occur when repetitive background objects are present in the scene. This is not always true in practice. Consider an indoor environment where the illumination comes from a fluorescence lamp. An example video sequence of such environment is taken from our lab, and 9 pixels picked up evenly from the scene are measured over time. Their RGB value distributions are drawn in Fig. 1(a). (All figures are also available through [1] for reference). Clearly shown from the figure, instead of 9 sphere like pixel distributions, the shapes of the pixel clusters are rather cylindrical. Pixel values tend to jump around more in one direction than another in the presence of illumination variations caused by the fluorescence lamp and camera jitter. This should be distinguished from the situation where one sphere distribution is moving slowly towards one direction due to slight daylight changes. Such a case is handled by updating the corresponding mean values in the original background model. Without an upper bound for the variance, the sphere describing the distribution tends to grow until it covers nearly every pixel in the most distributed direction, thus taking up a large space such that most of it does not belong to the distribution (sphere A in Fig. 1(b)). A simple solution to work around this problem is to set an upper limit for the variance, e.g. the maximum value of the variance in the least distributed direction. The result is multi-modal distributions represented as a series of smaller spheres (spheres B-E in the same figure). Although

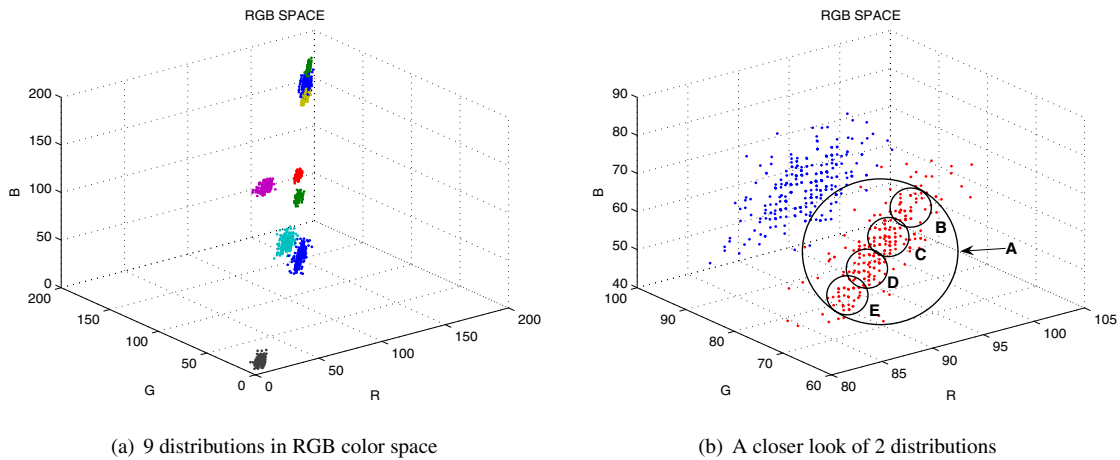


Figure 1. RGB color space

a background pixel distribution is modeled more precisely by such method, several Gaussian distributions are inferred which are hardware costly in terms of extra parameter updating and storage. In [10] D.Magee proposed a cylindrical model to address the issue, with primary axes of all distribution cylinders pointing at the origin. However, more parameters are needed for each cylindrical distribution than the spherical counterpart, let alone hardware costly computation needed to transform RGB value to cylindrical coordinates, e.g. division, square root. In addition, not every distribution cylinder is oriented to the origin, see the left middle distribution in Fig. 1(a).

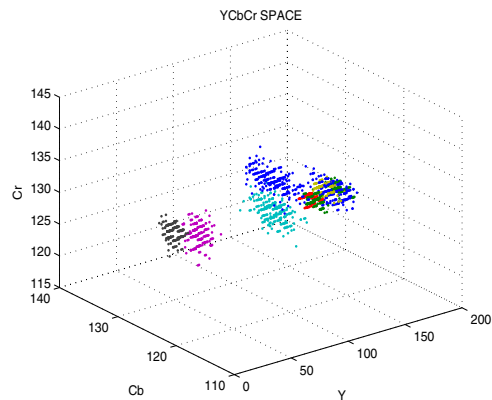


Figure 2. YCbCr color space

To be able to model background pixels using a single distribution but without much hardware overhead, color space transformation is employed in our implementation. By transforming RGB into YCbCr space, the correlation among different color coordinates in RGB space are mostly removed in YCbCr color space, resulting in nearly independent color components. With varying illumination environment, only the Y component (intensity) varies accordingly, leaving Cb and Cr components (chromaticity) more or less constant. In [7], such feature is utilized for shadow reduction. Consequently, values of three independent components of a pixel in YCbCr color space tends to spread equally. As shown in Fig. 2, most pixel distributions are transformed from cylinders back to spheres, capable of being modeled with a single distribution. The transformation from RGB to YCbCr is linear, and minor hardware overhead with only a few extra multipliers and adders are introduced, where multiplication with constance can be further utilized to reduce the hardware complexity.

### 3.2 Algorithm Simplifications

Two simplifications are made to the algorithm. As discussed in the previous section, setting an upper bound for the variance prevents a single distribution from growing too big. This also has a side effect that could result in more distributions but further simplified implementation. In the original algorithm specification, for the same reason that growing distribution will absorb more pixels, the weight of that distribution will soon dominate all others. To overcome this, all updated Gaussian distributions are sorted according to the ratio  $\omega/\sigma$  instead of  $\omega$  in [11]. In this way, the distribution with only dominant weight does not get to the top, identified as background distribution. With bounded variance, all distributions can now be sorted by their weights only, effectively eliminating division units in the implementation.

Another simplification is made in the process of fore-

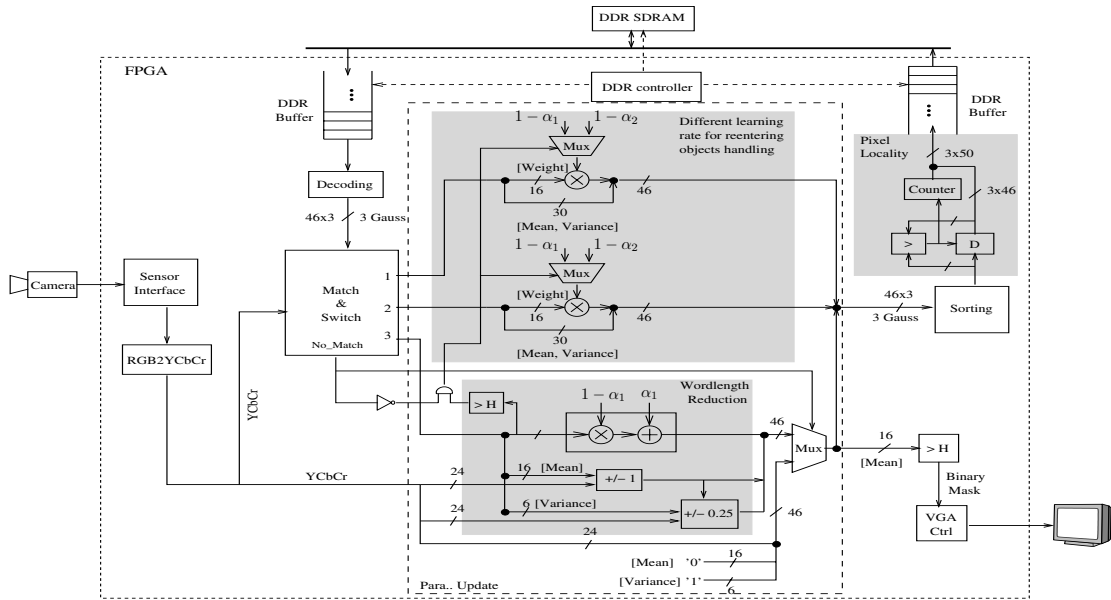


Figure 3. The system architecture of the segmentation unit.

ground/background detection. Instead of using equation 1, the determination can be made by checking the weight of each distribution only, since one pixel cluster will not spread out in several distributions by color space transformation to YCbCr. This results in automatic single or multi modal background model without having to adjust the value of H which is hard coded in the hardware.

### 3.3 Objects Reentering the Scene

With a slow learning factor, a foreground object that is standing still in the scene will finally become background after a relative long time, e.g. half a minute. This is a way for the background model to be adaptive to newly introduced background objects, e.g. a car enters a parking lot and stay there for a day. This could be problematic in some cases when people are standing still in front of a camera, e.g. a man who is filling up a form in front of a bank counter. Once he is background, around the same amount of time is required to fully remove him from the background distribution when he leaves. If the same person reenters the scene shortly afterwards, he will quickly be detected as background. To solve the problem, an approach with different learning factors is proposed for foreground/background switchings in forward and backward directions. Since the weight of each distribution works independently as indicated by the second simplification mentioned above, the forward and backward updating could be carried out at different speeds. As a result, a background object will turn into foreground much faster than a fixed foreground object takes to become background. However, this has to be dis-

tinguished with multi-modal situations, e.g. the leaf of a swaying tree will reenter the same position from time to time. A counter can be used to record the number of time a distribution switches between background and foreground. When a certain value is reached, e.g. 3-4, the distribution is regarded as multi-modal, thus foreground and background switchings are back to the same speed.

## 4 Hardware Architecture

To perform the algorithm with VGA resolution in real-time, a dedicated hardware architecture, with a streamlined data flow and memory bandwidth reduction schemes, is implemented to address the computation capacity and memory bandwidth bottlenecks. Algorithm modifications covered in previous sections are implemented with potential benefits on hardware efficiency and segmentation quality. This is a large improvement to the previous work [6], where only  $352 \times 288$  resolution is achieved without any memory reduction schemes and algorithm modifications. In this section, a brief overview of the system architecture of the segmentation unit is given, followed by detailed discussions on memory reduction schemes.

The system architecture is shown in Fig. 3 and works as follows: With each new incoming pixel from the camera converted to YCbCr components, a match is performed between the new pixel and a mixture of Gaussian that are read from the DDR buffer. A matched Gaussian is switched to the bottom(3 in the figure), where all of its parameters are to be updated. An unmatched Gaussian only needs to update its weight, where two learning factors are used to address

the issue of objects reentry. With a matched Gaussian being the background, the weights of unmatched Gaussian decrease quickly. A *no\_match* signal is asserted if a new pixel fits in no distributions, forcing the distribution on the bottom to be replaced by a new one. According to the simplifications made in previous sections, the foreground detection is achieved by simply comparing the weight of the distribution on the bottom with a predefined parameter  $H$ . All Gaussian parameters are sorted and stored in an DDR SDRAM, with manipulated data flow controlled by a DDR controller. With various buffers and pipelining, a sequence of binary data indicating background and foreground is streamed out to a monitor through a VGA controller.

#### 4.1 Wordlength Reduction

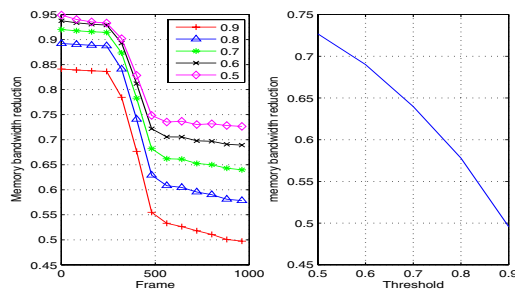
Slow background updating requires large dynamic range for each parameter in the distributions. This is due to the fact that parameter values are changed slightly between frames, but could accumulate over time. In this section, parameter wordlength reduction is investigated for potential memory bandwidth reduction.

According to Eqn. 3 and 4, the mean and variance of a Gaussian distribution is updated using a learning factor  $\rho$ . The difference of mean and variance between current and previous frame is derived from the equation as

$$\Delta_{\mu} = \mu_t - \mu_{t-1} = \rho(X_t - \mu_{t-1}), \quad (6)$$

$$\Delta_{\sigma^2} = \sigma_t^2 - \sigma_{t-1}^2 = \rho((X_t - \mu_t)^T(X_t - \mu_t) - \sigma_{t-1}^2). \quad (7)$$

Given a small value for  $\rho$ , e.g. 0.0001, a unit difference between the incoming pixel and the current mean value results in a value of 0.0001 for  $\Delta_{\mu}$ . To be able to record this slight change, 22 bits have to be used for the mean value, where 14 bits accounts for the fractional part. Less bits can be achieved by ignoring small deviations of the incoming pixel from current mean, while picking up only large ones. The extreme case is when only the largest deviation is picked, where the incoming pixel is in the range of 2.5 times standard deviation off the current mean. Larger than that, the incoming pixel will not match the current distribution. With an upper bound for the variance, e.g. 16, a maximum value of  $0.0001 \times 2.5 \times \sqrt{16} = 0.001$  is derived for  $\Delta_{\mu}$ , which can be represented by 10 bits. Using a wordlength lower than that, no changes would be recorded ever. In practice, the bits for fractional parts should be somewhere in between 10-14 bits. With similar calculations, 7-14 bits are obtained for the fractional parts of a variance. Together with 16 bits weight and integer parts of the mean and the variance, 81-100 bits are needed for a single Gaussian distribution. To reduce this number, a wordlength reduction scheme is proposed. From Eqn. 6, a small positive or negative number is derived depending on whether the incoming pixel is larger



**Figure 4. Memory bandwidth reduction over frames is shown to the left and memory bandwidth reduction versus different threshold is shown to the right.**

than the current mean. Instead of adding a small positive or negative fractional number to the current mean, a value of 1 or -1 is added. The overshooting caused by such coarse adjustment could be compensated by the update in the next frame, e.g. without illumination variation, the mean value will fluctuate with a magnitude of 1, which is negligible since Gaussian distribution is usually a sphere with a diameter of more than 10. In a relatively fast illumination varying environment, e.g. less than 25 RGB value changes in a second, fast adaptation to new lighting conditions is also enabled by adding or subtracting ones in consecutive frames. With coarse updating, only integers are needed for mean specification, which effectively reduce the wordlength from 18-22 down to 8 bits. Similar approach can be applied to the variance, resulting in a wordlength of 6 bits, where 2 bits account for fractional part. Together with the weight, the wordlength of a single Gaussian distribution can be reduced from 81-100 to only 44 bits, over 43% reduction is accomplished even compared to the extreme case in the normal updating scheme. In addition, less hardware complexity is achieved as a bonus since multiplication with the learning factor of  $\rho$  is no longer needed.

#### 4.2 Pixel Locality

In addition to wordlength reduction, a data compression scheme for further bandwidth reduction is proposed by utilizing pixel locality for Gaussian distributions in adjacent areas. We classify "similar" Gaussians in the following way: from the definition of a matching process, each Gaussian distribution can be simplified as a three dimensional cube in the YCbCr color space, where the center of the cube is composed of YCbCr mean values whereas the border to the center is specified by  $J$  times variance. One way to measure the similarity between two distributions is to check how much of the two cubes volume that overlap. If the over-



**Figure 5. The result before and after morphological filtering for different thresholds, (Left) original result, (Middle) with 0.8, and (Right) with 0.4 threshold.**

lap volume takes up certain percentage of both Gaussian cubes, they are regarded as "similar". The reason for such criteria lies in the fact that a pixel that matches to one distribution will most likely match to the other, if they have enough overlapping volume. The percentage is a threshold parameter that can be set to different values among different simulations.

In the architecture, two similar distributions are treated as equivalent. By only saving non overlapping distributions together with the number of equivalent succeeding distributions, memory bandwidth is reduced. Various threshold values are selected to evaluate the efficiency for memory bandwidth reduction. With a low threshold value where less overlapping Gaussians are regarded as the same, more savings could be achieved. However, more noise is generated due to increasing mismatches in the matching block. Fortunately, such noise is found non-accumulating and therefore can be reduced by later morphological filtering[4]. Fig. 4 shows the memory bandwidth savings over frames with various threshold values. From the figure, memory bandwidth savings tends to stabilize (around 60%) after initialization. The quality of segmentation results before and after morphology is shown in Fig. 5, where it is clear that memory reduction comes at the cost of segmentation quality. Too low threshold value results in clustered noises that would not be filtered out by morphological filtering. In this implementation, a threshold value of 0.8 is selected, combined with wordlength reduction scheme, a memory bandwidth reduction of over 70% is accomplished.

## 5 Results

The system is implemented on a Xilinx VirtexIIpro vp30 FPGA development board. A KODAK KAC-9648 CMOS sensor is used to capture color images at 25 fps with VGA resolution. Real time segmentation performance is achieved on video sequences with 3 Gaussian distributions per pixel

at an operating frequency of 100 MHz. The issue regarding objects reentrance is much improved. With the proposed memory reduction schemes, off-chip memory bandwidth is reduced from 576 MB/s to 170 MB/s.

## 6 Conclusions

Real-time video segmentation with VGA resolution can be achieved by proposed joint memory reduction scheme. Substantial memory bandwidth reduction comes at the cost of segmentation quality. This can be minimized by careful tradeoffs and specific morphology post processing. Algorithm modifications are of great importance for the efficiency of the hardware implementation. By utilizing different color space, several simplifications are made that results in great hardware savings. The algorithm is implemented on an Xilinx VirtexIIpro vp30 FPGA development board, capable of video segmentation at 25 FPS with VGA resolution.

## References

- [1] [http://www.es.lth.se/home/htj/avss\\_figs](http://www.es.lth.se/home/htj/avss_figs).
- [2] D. Gao *et al.* Adaptive background estimation for real-time traffic monitoring. In *Proc. ITSC*, 2001.
- [3] M. Harville *et al.* Adaptive video background modeling using color and depth. In *Proc. ICIP*, 2001.
- [4] H. Hedberg *et al.* Low complexity architecture for binary image erosion and dilation using structuring element decomposition. In *Proc. ISCAS*, 2005.
- [5] O. Javed *et al.* A hierarchical approach to robust background subtraction using color and gradient information. In *Proc. Workshop on Motion and Video Computing*, 2002.
- [6] H. Jiang *et al.* Hardware accelerator design for video segmentation with multi-modal background modelling. In *Proc. ISCAS*, 2005.
- [7] F. Kristensen *et al.* Background segmentation beyond rgb. In *ACCV*, 2006.
- [8] L. Li *et al.* Foreground object detection in changing background based on color co-occurrence statistics. In *Proc. 6th IEEE WS on Applications of Computer Vision*, 2002.
- [9] N. Li *et al.* Real-time video object segmentation using hsv space. In *Proc. ICIP*, 2002.
- [10] D. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, (22):143–145, 2004.
- [11] C. Stauffer *et al.* Adaptive background mixture models for real-time tracking. In *IEEE CVPR*, 1999.
- [12] K. Toyama *et al.* Wallflower: principles and practice of background maintenance. In *ICCV*, 1999.
- [13] Y. Tsaig *et al.* Automatic segmentation of moving objects in video sequences: A region labeling approach. *IEEE tran. circuits and systems for video technology*, July 2002.
- [14] H. Wang *et al.* A re-evaluation of mixture-of-gaussian background modeling. In *Proc. ICASP*, 2005.
- [15] Y. Xu *et al.* Illumination invariant motion estimation for video segmentation. In *Proc. IEEE. ICIP*, 2004.