# A Framework for High-Level Feedback to Adaptive, Per-Pixel, Mixture-of-Gaussian Background Models

Michael Harville

Hewlett-Packard Laboratories
1501 Page Mill Rd. ms 3U-4, Palo Alto, CA 94304, United States
harville@hpl.hp.com

**Abstract.** Time-Adaptive, Per-Pixel Mixtures Of Gaussians (TAPPMOGs) have recently become a popular choice for robust modeling and removal of complex and changing backgrounds at the pixel level. However, TAPPMOG-based methods cannot easily be made to model dynamic backgrounds with highly complex appearance, or to adapt promptly to sudden "uninteresting" scene changes such as the re-positioning of a static object or the turning on of a light, without further undermining their ability to segment foreground objects, such as people, where they occlude the background for too long. To alleviate tradeoffs such as these, and, more broadly, to allow TAPPMOG segmentation results to be tailored to the specific needs of an application, we introduce a general framework for guiding pixel-level TAPPMOG evolution with feedback from "high-level" modules. Each such module can use pixel-wise maps of positive and negative feedback to attempt to impress upon the TAPPMOG some definition of foreground that is best expressed through "higher-level" primitives such as image region properties or semantics of objects and events. By pooling the foreground error corrections of many high-level modules into a shared, pixel-level TAPPMOG model in this way, we improve the quality of the foreground segmentation and the performance of all modules that make use of it. We show an example of using this framework with a TAPPMOG method and high-level modules that all rely on dense depth data from a stereo camera.

## 1 Introduction and Motivation

Many computer vision and video processing applications, in domains ranging from surveillance to human-computer interface to video compression, rely heavily on an early step, often referred to as "foreground segmentation" or "background removal", that attempts to separate novel or dynamic objects in the scene ("foreground") from what is normally observed ("background"). Recently, methods employing Time-Adaptive, Per-Pixel Mixtures Of Gaussians (TAPPMOGs) have become a popular choice for modeling scene backgrounds at the pixel level [4–6, 8, 10, 12, 15, 16, 20]. In these methods, the time series of observations at a given image pixel is treated as independent of that for all other pixels, and is modeled using a mixture of Gaussians. The per-pixel models are updated as new

observations are obtained, with older observations losing influence over time. At each time step, a subset of the Gaussians in each per-pixel model is selected as representative of the scene background, and current observations that are not well-modeled by those Gaussians are designated as foreground.

We refer to a TAPPMOG as a "pixel-level" foreground segmentation method because it classifies the current scene image as foreground or background at each pixel using only image information that has been collected at or near that pixel over time. Many other pixel-level foreground segmentation techniques exist, but TAPPMOG methods have gained favor because, unlike most others, they possess the following combination of attributes:

- TAPPMOGs slowly adapt their background model to persistent scene appearance modifications such as the insertion or removal of an object, the moving of an object such as a rolling swivel chair from one static position to another, and the change in global illumination that occurs when a light is turned on or when the sun passes behind clouds. While such changes may be of temporary interest, in some applications and hence may be an acceptable part of the foreground for some period of time, it is usually not desirable to pay special attention to them indefinitely.
- TAPPMOGs are relatively proficient at modeling the largely repetitive scene appearance changes associated with dynamic objects such as moving foliage, ocean waves, or rotating fans. Although many applications are interested in dynamic objects, most would also like to disregard those whose behavior, when observed over long periods of time, does not seem to vary much.
- TAPPMOGs are suitable for real-time software implementation. For applications such as interactive human-computer interfaces and long-running activity-monitoring systems, it is critical that the foreground segmentation process does not require excessive computational resources or time.

Other proposed methods of pixel-level background modeling, such as ones based on Wiener prediction filtering [17] and Bayesian decision-making with normalized histogram probability representations [11], can be argued to also possess the above attributes. Little work has been done in comparing the pixel-level results of such methods with those of TAPPMOGs for the same input video sequences, but informal comparison of published results suggests that none of the alternatives perform significantly better than the best TAPPMOG variants.

Nevertheless, the foreground produced by TAPPMOGs still falls far short of the ideal for many applications. TAPPMOGs struggle with performance tradeoffs in two areas in particular:

**Learning rate:** The speed with which a TAPPMOG's Gaussian mixtures begin to reflect more recent observations in favor of older ones is controlled by a parameter called the "learning rate". A high learning rate allows the TAPPMOG to adapt its background model more quickly to sudden, persistent scene appearance changes, such as those caused by the turning on of a light or the re-positioning of a chair, that are foreground distractions to applications concerned with analyzing the activities of people, automobiles, animals, or other

dynamic, purposeful objects. Such distractions can severely compromise overall system performance if they are not accurately detected and compensated for by further - and often computationally expensive - processing. Unfortunately, a high learning rate also causes people who do not move enough to fade more quickly into the background. When a person is expected to remain in roughly the same location relative to the camera for extended time periods, as is the case in many vision-based human-computer interface applications, background adaptation is an especially challenging problem. Also, at a "high-traffic" scene location, such as at a busy retail store doorway, where many different foreground objects (in this case, people) frequently pass, a high learning rate will cause the background model to degrade more quickly into a reflection of some average properties of the passing foreground objects.

**Background model inclusivity:** Although we may wish to treat as background those dynamic objects, such as foliage in wind, that exhibit somewhat recurrent patterns of motion, it becomes increasingly difficult to model these objects as their motion patterns include a greater number of significant frequency components, or as their visual textures, even when the objects are static, grow more complex. In such cases, TAPPMOGs usually require more Gaussians to adequately model the distribution of observations produced by the object at a single pixel over time. As more Gaussians are included in the background model, however, it becomes more likely that some representing the desired foreground will also be selected, thereby resulting in erroneous foreground omissions. For instance, it is difficult to determine, using only local image statistics, whether a small number of observations, modeled by one Gaussian, correspond to a foreground person who passed by quickly or to an occasionally recurring motion dynamic of a background object.

One might attempt to address these tradeoffs through modifications of the basic TAPPMOG methods, or by choosing an entirely different framework for pixel-level background modeling. It seems likely, however, that TAPPMOGs already approach the limit of what can reasonably be accomplished through consideration of local information alone. Each of a TAPPMOG's per-pixel Gaussian mixtures can, with sufficient components, accurately describe the appearance statistics of arbitrarily complex backgrounds, and can be modified very quickly or very slowly, as needed, by simple, well-principled processes. What is lacking is more intelligent guidance of the mixtures' evolution, and it would seem wise to drive this guidance with analysis that operates on "higher" levels of information, such as image regions or entire frames, or object and event semantics. Put another way, in applications for which the definition of the ideal foreground segmentation depends on concepts such as region cohesiveness, frame-wide illumination changes, or object classification, we should not expect to produce this segmentation by relying solely on methods that consider each pixel in isolation.

In this paper, therefore, we extend TAPPMOGs in a general way to incorporate corrective guidance from analysis concerned with higher level primitives such as image regions, image frames, or object semantics. This "high-level feedback" allows for many significant improvements in background modeling, including:

1. The rapid incorporation into the background model of appearance changes associated with events that are uninteresting to the application. The definition of "uninteresting" can be application-dependent: some may want to ignore rapid, global illumination changes, while others may want to ignore an automobile after its driver has stopped and exited it. Almost any such definition can be accomodated through feedback from an appropriate high-level module.
2. Good segmentation of foreground objects of interest, even when they stop moving or occlude the background for indefinitely long times. Again, the definition of what foreground is "interesting" may be application-dependent, and most definitions can be serviced with a proper choice of high-level feedback.
3. Much better exclusion from the foreground of dynamic objects that move in repetitive ways. Higher levels of processing can detect the basic TAPPMOG methods' failures to model such objects, and then feed back information on how to adjust the pixel-level model to prevent re-occurrence of the problems.

More broadly, feedback enables higher levels of processing to tailor the general-purpose pixel-level segmentation produced by TAPPMOGs to the specific definition of "foreground" pertinent to an application. This, in turn, should improve the performance of all high-level modules that depend on the TAPPMOG segmentation, whether or not those modules are providing feedback.

A number of other methods, including those of [2, 10, 13, 17], can be interpreted as using high-level feedback to guide pixel-level foreground segmentation. All of these methods, however, modify their pixel-level background modeling only in response to specific types of high-level processing, such as the detection of people or illumination changes. None describe how to generalize the feedback influence to arbitrary forms of high-level analysis. Furthermore, with the exception of [10, 17], they use pixel-level statistical models that are too simple to represent complex, dynamic backgrounds. We believe our method represents a novel and appealing combination of powerful, per-pixel background modeling with a flexible, general-purpose mechanism for enhancement via high-level input.

In Section 2, we describe a particular variant of the basic TAPPMOG method in greater detail, and provide examples of its successes and shortcomings on a challenging test sequence. In Section 3, we discuss our framework for augmenting basic TAPPMOG methods with corrective feedback from high-level processing modules. Finally, in Section 4, we illustrate how several specific types of feedback can dramatically improve pixel-level foreground segmentation results.

## 2 Background removal at the pixel level

Harville et. al. [8] have recently described a TAPPMOG-based method that, under the assumption of a static camera, seeks to segment novel objects in a scene, as well as non-novel objects that begin behaving in novel ways. It attempts to ignore phenomena such as illumination changes, shadows, inter-reflections, persistent scene modifications, and dynamic backgrounds. This definition of "foreground" is consistent with a wide variety of vision applications, and hence the method is of broad utility. By modeling the scene in the combined feature space of per-pixel depth and luminance-normalized color, the method better adheres to

this definition than most other choices for pixel-level background removal, and it better avoids foreground omissions due to color "camouflage" (similarity of appearance between a novel object and the background behind it). The method's reliance on dense (per-pixel) depth has become more practical in recent years as hardware and software for stereo cameras has become much faster and cheaper [9, 14, 18].

We briefly review the method of [8] here, and show some example results. The results contain many successes, but also the types of failures discussed in Section 1 that motivate our use of high-level feedback.

### 2.1 On-line clustering of observations

The pixel-level background modeling method of [8] regards the time series of observations at each pixel as an independent statistical process. Each pixel observation consists of a color and a depth measurement. Color is represented in the YUV space, which allows for separation of luminance and chroma. Depth measurements, denoted as $D$, are produced by a real-time stereo camera implementation, but could also be computed by methods based on active illumination, lidar, or other means. The observation at pixel $i$ at time $t$ can be written as $\boldsymbol{X}_{i,t} = [\, Y_{i,t}\ U_{i,t}\ V_{i,t}\ D_{i,t}\,]$.

The history of observations at a given pixel, $[\boldsymbol{X}_{i,1}, \ldots, \boldsymbol{X}_{i,t-1}]$, is modeled by a mixture of $K$ Gaussian distributions. $K$ is the same for all pixels, typically in the range of 3 to 5. The probability of the current observation at pixel $i$, given the model built from observations until the prior time step, can be estimated as

$$P(\boldsymbol{X}_{i,t}|\boldsymbol{X}_{i,1}, \ldots, \boldsymbol{X}_{i,t-1}) = \sum_{k=1}^{K} w_{i,t-1,k} * \eta(\boldsymbol{X}_{i,t}, \boldsymbol{\mu}_{i,t-1,k}, \Sigma_{i,t-1,k}) \qquad (1)$$

where $\eta$ is a Gaussian probability density function, where $w_{i,t-1,k}$ is the weight associated with the $k^{th}$ Gaussian in the mixture at time $t-1$, , and where $\boldsymbol{\mu}_{i,t-1,k}$ and $\Sigma_{i,t-1,k}$ are the mean $YUVD$ vector and covariance matrix of this $k^{th}$ Gaussian. The weights $w_{i,t-1,k}$ indicate the relative proportions of past observations modeled by each Gaussian. A diagonal covariance matrix is used. For notational simplicity, we will denote the $kth$ Gaussian of a mixture as $\eta_k$.

To update a pixel's mixture model as new observations are obtained over time, an on-line K-means approximation similar to that of [16] is used. When a new observation $\boldsymbol{X}_{i,t}$ at a given pixel is received, an attempt is made to find a match between it and one of the Gaussians $\eta_k$ for that pixel. If a matching $\eta_k$ is found, its parameters are adapted using the current observation; if not, one of the Gaussians is replaced with a new one that represents the current observation. The matching process is carried out by first sorting the Gaussians in a mixture in order of decreasing weight/variance, and then selecting as a match the first $\eta_k$ whose mean is sufficiently near $\boldsymbol{X}_{i,t}$. A match between $\boldsymbol{X}_{i,t}$ and $\eta_k$ is allowed if each squared difference between corresponding components of $\boldsymbol{X}_{i,t}$ and the mean $\boldsymbol{\mu}_{i,t-1,k}$ of $\eta_k$ is less than some small multiple $\beta$ of the corresponding $\eta_k$ component variance. The parameter $\beta$ is typically chosen to be about 2.5, so that the boundary of the matching zone in YUVD-space for

$\eta_k$ encompasses over 95% of the data points that would be drawn from the true Gaussian probability density.

This basic matching method is modified, however, to account for the possibility of unreliable, or "missing", chroma or depth data. At low luminance, the chroma components (U and V) of the color representation become unstable, so chroma information is not used in comparing the current observation $\boldsymbol{X}_{i,t}$ with the mean of Gaussian $\eta_k$ when the luminance of either falls below a threshold $Y_{min}$. Similarly, because stereo depth computation relies on finding small area correspondences between image pairs, it does not produce reliable measurements in regions of little visual texture and in regions, often near depth discontinuities in the scene, that are visible in one image but not the other. Most stereo depth implementations attempt to detect such cases and label them with one or more special values, which can be denoted collectively as *invalid*. If the depth of $\boldsymbol{X}_{i,t}$ is *invalid*, or if too many (more than some fraction $\rho$) of the observations that contributed to the building of $\eta_k$, as described below, had *invalid* depth, depth is omitted in comparing $\boldsymbol{X}_{i,t}$ and $\eta_k$.

If no mixture Gaussian is found to match $\boldsymbol{X}_{i,t}$, the Gaussian ranked last in weight/variance is replaced by a new one with mean equal to $\boldsymbol{X}_{i,t}$, a high variance, and a low weight. Otherwise, the parameters of the matching Gaussian $\eta_k$ are recursively adapted toward $\boldsymbol{X}_{i,t}$. The mean is updated as follows:

$$\boldsymbol{\mu}_{i,t,k} = (1-\alpha)\boldsymbol{\mu}_{i,t-1,k} + \alpha \boldsymbol{X}_{i,t} \tag{2}$$

The variances are updated analogously, using the squared differences between corresponding components of $\boldsymbol{X}_{i,t}$ and $\boldsymbol{\mu}_{i,t,k}$. Variances are not allowed to fall below some minimum value, so that matching does not become unstable in scene regions that are static for long time periods. The parameter $\alpha$ can be interpreted as the learning rate discussed in Section 1: as $\alpha$ is made smaller, the parameters of $\eta_k$ will be perturbed toward new observations in smaller incremental steps.

The weights for all Gaussians are updated according to

$$w_{i,t,k} = (1-\alpha)w_{i,t-1,k} + \alpha \mathcal{M}_{i,t,k} \tag{3}$$

$\mathcal{M}_{i,t,k}$ is 1 (true) for the $\eta_k$ that matched the observation and 0 (false) for all others, so (3) causes the weight of the matched $\eta_k$ to increase and all other weights to decay.

Shadows and other illumination effects are specifically addressed in two ways. First, the variances for the luminance components of the Gaussians are not allowed to decrease below a substantial floor level. Since the luminance component of an object's apparent color is typically more impacted than its chroma by shadows and other illumination changes, representation of this luminance with a higher variance allows it to be matched to current observations of the object under a wider variety of lighting conditions. Second, where the current depth observation and the depth statistics of some Gaussian $\eta_k$ are both reliable and are a match, the color matching criterion is relaxed by increasing the matching tolerance $\beta$. This helps in ignoring phenomena such as strong shadows, for which the shape of the background matches the current observations, but for which the color is significantly different. This second technique also helps model dynamic

background objects, like video displays and moving foliage, whose color is highly variable but whose shape, at least at a coarse level, remains relatively constant.

Finally, the learning rate $\alpha$ is greatly reduced at all pixels at which a scene "activity" level $A$ is above a threshold $H$. $A$ is computed independently at each pixel as a temporally-smoothed inter-frame luminance difference:

$$A_{i,t,k} = (1 - \lambda)A_{i,t-1,k} + \lambda \left| Y_{i,t} - Y_{i,t-1} \right| \tag{4}$$

This helps reduce the influence of dynamic foreground objects such as people on the model of the observation history, without slowing adaptation of the model to persistent changes such as a moved chair or increased lighting.

### 2.2 Background model estimation and foreground segmentation

At each time step, one or more of the Gaussians in each per-pixel mixture are selected as the background model, while any others are taken to represent foreground. The current observation at a pixel is designated as foreground if it was not matched to any of the $\eta_k$ in the pixel's current background model.

Background Gaussians are selected at each pixel according to two criteria. First, among the Gaussians with reliable depth statistics (those for which the fraction of observations modeled that have valid depth exceeds a threshold $\rho$) and whose normalized weight $w'_k = w_k / \left(\sum_k w_k\right)$ exceeds a low threshold $T_D$, the $\eta_k$ with the largest depth mean is selected. This criterion is based on the idea that, in general, one does not expect to be able to see through the background. The threshold $T_D$ discourages the selection of a background Gaussian $\eta_k$ based on spurious or transient observations. $T_D$ is typically set around 0.1 to 0.2, so that an $\eta_k$ representing the true background may be selected even when this background is usually occluded or has not been visible for an extended time.

At pixels where the true background usually does not produce reliable depth data, no Gaussian will satisfy the above criterion. For correlation-based stereo, this commonly occurs for walls and floors in the scene, which tend to have low visual texture. Furthermore, for dynamic background objects, the background is often best represented by multiple Gaussians, not just the one selected by the above criterion. The second criterion for selecting background Gaussians addresses both of these issues by adding $\eta_k$, in order of decreasing weight/variance, to the background model until the total weight of all selected Gaussians exceeds a second threshold $T$. This second criterion is typically the sole criterion used by TAPPMOG methods that have no access to depth data, and it favors the selection of Gaussians representing consistent, frequently observed modes of scene appearance. The parameter $T$ helps determine the level of "background model inclusivity" discussed in Section 1: as $T$ is increased, more of a pixel's observation history Gaussians are likely to be selected as background.

### 2.3 Experimental results

The performance of the pixel-level foreground segmentation method of [8] was evaluated on a challenging color-and-depth test sequence captured by the Tyzx stereo camera head [18]. The camera head makes use of special-purpose hardware, based on [19], for computing depth by correlation of Census-transformed images [21]. The camera can produce spatially-registered, 320x240-resolution


**Fig. 1.** Comparison of results for the pixel-level segmentation method of [8] **(bottom row)** to those for a common alternative **(middle row)** that uses an RGB input space rather than YUVD. Input color (**top row**) and depth (not shown) were captured using the Tyzx real-time stereo camera head [18]. Locations of several of the test sequence challenges are indicated with text in middle row images. Parameter values were $K = 4, \alpha = 0.0006, \beta = 2.5, \rho = 0.2, Y_{min} = 16, T = 0.4, T_D = 0.2, \lambda = 0.09, H = 5$. Small, isolated foreground regions, and small holes within foreground, were removed by applying an area threshold to the results of connected-components analysis.

color and depth images at rates up to 60Hz; the test imagery was saved to files at a rate of 15Hz.

The test sequence is 10 minutes long, with no image devoid of "foreground" people. It contains several dynamic background objects, namely several video displays (toward the upper left of the images) and a sign rotating about a vertical axis at about 0.5Hz (upper middle of images, sitting on oval-shaped table). During the first half of the sequence, two displays ("display1" and "display2") are active and one ("display3") is off, while two people walk around the room. Near the sequence midpoint, the chair in the lower left of the image is moved to a new floor position, "display2" is switched off, "display3" is switched on, and several more people enter the scene. One of these people stands in the middle back part of the room for the rest of the sequence, sometimes shifting his weight or moving his arms. The other new people walk around continuously in the lower right part of the images, creating a "high-traffic" area.

Figure 1 compares results for the method of [8] with those for a more standard TAPPMOG technique that uses an RGB input space rather than YUVD. The depth-aided method better eliminates shadows, and better excludes the video displays and rotating sign from the foreground. Segmentation of people in the high-traffic area is improved because Gaussians representing the floor are usually selected as background, on the basis of depth, despite the more frequently observed, but closer, people. Finally, depth allows better segmentation of people colored like the background, resulting in fewer foreground holes due to color camouflage.

Still, the results of this pixel-level method are far from perfect. Although the video displays and rotating sign do not appear in the result frames in Figure 1, they fail to be excluded from the foreground in a significant fraction of other frames. The relatively static person at the back of the room contains substantial foreground holes after he been there for about 3 minutes. It is difficult to extend this time without further compromising the modeling of the dynamic background objects in the scene. Adaptation to the moving of the chair requires about 2 minutes, and cannot be shortened without causing all of the above problems to worsen. A rapid illumination change would cause almost all of the scene to appear as foreground until adaptation occurs. Hence, there remains great motivation for improvement of the method, and we believe this is best achieved by incorporating, as presented in the following sections, analysis that is concerned with more than just spatially local, per-pixel statistics.

## 3 Feedback for background correction

We extend the TAPPMOG modeling framework to make use of a wide variety of feedback computed by modules that consider image regions or frames, classify objects, or analyze other scene properties above the per-pixel level. Each module computing this feedback must satisfy two requirements. First, it must have some ability to detect foreground segmentation successes and/or failures according to some definition in terms of "higher" concepts than pixel-level statistics. The module's detection ability does not have to be perfect, and the module's definition of foreground does not need to encompass the entire application demands. Second, the module must be able to provide maps that designate which pixels in a given input frame are responsible for cases that satisfy this definition.

We make use of two types of feedback: **1) positive** feedback, which serves to enhance correct foreground segmentations, and **2) negative** feedback, which aims to adjust the pixel-level background model in order to prevent the reoccurrence of detected foreground mistakes. The feedback interface between the TAPPMOG background model and the higher levels consists of two bitmaps, representing pixels where positive and negative feedback, respectively, should be applied. We denote these maps as $\mathcal{P}$ and $\mathcal{N}$, respectively.

In a system that uses multiple high-level modules to generate feedback maps, we will need some mechanism for combining these maps and resolving conflicts between them in order to produce the two bitmaps. We would like to take an approach that allows strong positive evidence to override negative evidence (or vice versa), permits individual modules to generate both positive and negative feedback, enables mutiple forms of relatively weak feedback to support each other when none are convincing in isolation, and allows us to refine the feedback bitmaps by cancelling out portions of them where conflicting information exists.

We achieve this by implementing each high-level module so that it generates not a feedback bitmap, but rather a map of real numbers, where positive numbers reflect confidence that the segmented pixel should be part of the foreground, and negative numbers reflect the opposite. We then pixel-wise add together the feedback maps generated by all high-level modules, and threshold the summation map twice at 1 and -1 to produce $\mathcal{P}$ and $\mathcal{N}$. This method allows us to factor

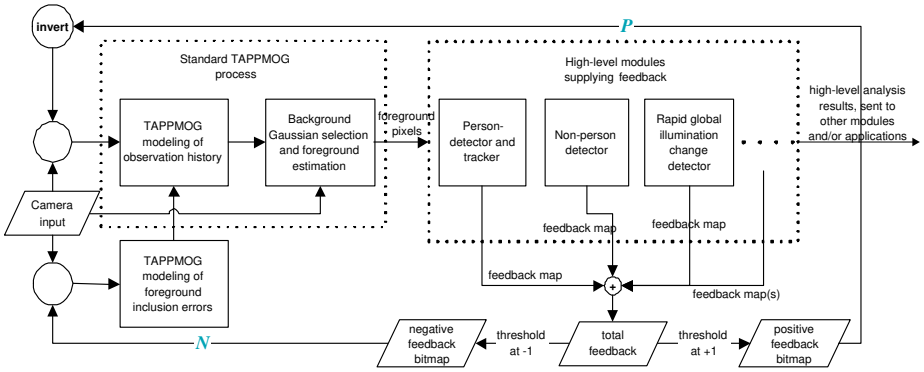


**Fig. 2.** Overview of framework for guiding TAPPMOG with high-level feedback.

the relative confidences associated with various high-level decisions into the final choice of corrective action to take at each pixel.

The methods by which positive and negative feedback influence pixel-level background modeling are different, and are discussed in the subsections below. These methods are suitable for use not just with the TAPPMOG modeling scheme described in Section 2, but with most other TAPPMOG-based methods. Figure 2 summarizes the feedback process.

### 3.1 Positive feedback

The goal of positive feedback is to prevent observations associated with correctly segmented foreground from being incorporated into the pixel-level background model. In the test sequence described in section 2.3, the two cases for which this would be most helpful are those of the relatively static person toward the back of the room, and the high-traffic area in the lower right of the frame.

Our implementation of positive feedback is quite simple. First, one or more high-level modules detect correct foreground segmentation results, by one or more definitions, and contribute positive feedback at all pixels responsible for these results. This feedback propagates to the bitmap $\mathcal{P}$ as described above. Next, for all pixels at which $\mathcal{P} = 1$, we do not use the current pixel observation to update the Gaussian mixture model of the observation history. This results in no change in the background model at those pixels, and prevents the foreground objects from becoming part of the background over time.

It is relatively unimportant that $\mathcal{P}$ be precise at the pixel level. Pixels mistakenly omitted from $\mathcal{P}$ cause some portions of true foreground to be incorporated into the observation history model. Extra pixels included in $\mathcal{P}$ cause the true background to be learned slightly more slowly. In both cases, the same error must repeat many times before the effect is significant.

### 3.2 Negative feedback

An erroneous inclusion in the segmented foreground is, by definition, something that we would prefer to be well-described by the background model. The goal of negative feedback, therefore, is to adjust the background model so that it better describes such errors, without disrupting its ability to describe other aspects of the background. We achieve this with a combination of two processes:

**TAPPMOG error modeling:** We model the distribution of observations associated with foreground errors at each pixel using almost exactly the same TAPPMOG process, described in Section 2.1, that is employed for modeling the full observation history. We denote the two TAPPMOG models of the observation history and the foreground errors as $\mathcal{O}$ and $\mathcal{E}$, respectively. As described above, $\mathcal{O}$ receives the camera input directly, except where positive feedback, as indicated by $\mathcal{P}$, occurs. $\mathcal{E}$, on the other hand, receives only the portions of the camera input thought to be associated with foreground inclusion errors (background modeling failures), as indicated by the negative feedback bitmap $\mathcal{N}$. No update of $\mathcal{E}$ occurs at pixels for which $\mathcal{N}$ contains a zero. A TAPPMOG is used to model foreground errors because the distribution of observations associated with errors at a given pixel can, at worst, be as complex as observation distributions for highly dynamic, variable backgrounds.

**Model merging:** We periodically merge the error model $\mathcal{E}$ into the observation history model $\mathcal{O}$, in the hope that changes in $\mathcal{O}$ will propagate into the subset of Gaussians selected as background. We merge, rather than replace $\mathcal{O}$ with $\mathcal{E}$, because portions of $\mathcal{O}$ may still be accurate and useful. This is particularly true when the errors result from inadequate modeling of dynamic background objects. In general, errors occur because too little evidence in the observation history supports building an accurate model of them with sufficient weight to be chosen as background. Hence, the merging process aims to boost the relative proportion of evidence corresponding to things that were incorrectly omitted from the background model, without obliterating other highly-weighted evidence.

Because the maximum complexities of what may be modeled by $\mathcal{O}$ and $\mathcal{E}$ are similar, we use mostly the same parameters for each. The main exception is that we use a higher learning rate, which we denote as $\alpha_e$, for $\mathcal{E}$. Because error examples may be presented to this TAPPMOG rather infrequently, error model means and variances might converge very slowly if we were to use the same learning rate as for $\mathcal{O}$. In addition, from equation (3), we see that a higher learning rate will cause the weights associated with $\mathcal{E}$ to increase more quickly. When $\mathcal{O}$ and $\mathcal{E}$ are merged as described below, these higher weights help compensate for the under-representation of the errors in the observation history.

While update of $\mathcal{E}$ with error information is done for every frame, we merge $\mathcal{E}$ with $\mathcal{O}$ at a low rate $\theta$ in the range of 0.2-1.0Hz, to avoid excessive computational cost. When merging the Gaussian mixtures of $\mathcal{O}$ and $\mathcal{E}$ at a particular pixel, we do not simply make a new mixture that has one mode for each of the modes in the two original mixtures, since this would cause the complexity of $\mathcal{O}$ to grow without bound. Instead, we seek to keep the number of Gaussians at each pixel in $\mathcal{O}$ at or below the limit $K$. A well-principled way to merge the two mixtures under this constraint would be to convert each to a histogram representation, and then use an iterative Expectation-Maximization method to fit a mixture of $K$ Gaussians to the sum of the two histograms. This would be a rather costly procedure, particularly as the dimensionality of the observation feature space increases, so we instead adopt the following, more approximate approach that allows for real-time implementation:

1. Merging is performed at a pixel only if the total weight of the error Gaussians at that pixel exceeds a threshold $\kappa$. As $\kappa$ is decreased, less error evidence is needed to trigger a modification of $\mathcal{O}$, and each modification will have smaller impact. For larger $\kappa$, errors tend to be better modeled before they are merged into $\mathcal{O}$, and individual merges occur less frequently but are more influential.
2. To merge $\mathcal{E}$ and $\mathcal{O}$ at a pixel, we first attempt to find observation history Gaussians $\eta_k^o$ whose variance parameters may be increased to include the spaces spanned by nearby error model Gaussians $\eta_j^e$. Expansion of $\eta_k^o$ to include $\eta_j^e$ is permitted if their means are separated by less than $\beta$ times the sum of their standard deviations. A single $\eta_k^o$ may be expanded to include more than one $\eta_j^e$. When merging two Gaussians, we add their weights, set the new mean equal to their weighted sum, and select the minimum variance large enough to cause all points that would have matched either one of the original Gaussians to also match the merge result.
3. If an error Gaussian $\eta_j^e$ is not near any $\eta_k^o$, we substitute $\eta_j^e$ for the "poorest" observation model Gaussian $\eta_k^o$ - namely that with the lowest weight/variance ratio - provided that we would not be replacing one Gaussian with another supported by far less evidence. This latter criterion is enforced by ensuring that the ratio of the weight of $\eta_j^e$ to that of $\eta_k^o$ is above a threshold **minweightratio**.
4. Gaussians $\eta_j^e$ that are merged in either of the prior two steps are removed from $\mathcal{E}$, while the others remain in it and will continue to evolve over time.
5. After merging is completed, we normalize the weights of the observation Gaussians so that they add up to their sum prior to the merge. This prevents the scale of these weights from fluctuating depending on the rate at which segmentation errors are occurring.
6. To prevent the accumulation of errors over arbitrarily long times, all weights for error Gaussians not merged into the observation history model are decayed by a multiplicative factor $0 < \tau < 1$. Values of $\tau$ at the higher end of this range allow for errors to be accumulated over longer time periods without being disregarded as noise.

Examination of the negative feedback process reveals that it is relatively unimportant for $\mathcal{N}$ to be precisely correct at the pixel level. If $\mathcal{N}$ extends slightly beyond the true bounds of some erroneous foreground inclusion, the result will usually be the addition to $\mathcal{E}$ of further evidence to support the current background model. If $\mathcal{N}$ fails to include some of the pixels associated with a true error, $\mathcal{E}$ will just build up a little more slowly at those locations.

## 4 Examples of feedback generation and usage

In this section, we illustrate the beneficial effects of using the feedback correction framework of Section 3 to connect two types of "high-level" analysis modules to the TAPPMOG method of Section 2. Since these are just two of the infinite number of high-level modules that could be employed in our framework, and since the emphasis of this paper is on the feedback itself, and not on the high-level

analysis that generates it, we omit most of the algorithmic details of the high-level modules. However, we note that, like the TAPPMOG method of Section 2, both modules make interesting use of dense depth imagery.

One of these modules is a person detection and tracking method. We use the results of this method to produce feedback that enhances the TAPPMOG's segmentation of people, and helps it to ignore all else. Positive feedback is generated for image regions where the person tracker believes people are present, while all other foreground pixels are assumed not to pertain to people, and are associated with negative feedback. Our person tracking method is described in [7], and has similarities to the methods of [1, 3, 18]. In brief, it uses depth data to create overhead, "plan-view" images of the foreground produced by the TAPPMOG of Section 2, and then uses templates and Kalman filtering to detect and track people in these images. For each tracked person, positive feedback (with a value of 1) is generated at all pixels within the camera-view bounding box of the set of pixels that contributed to the person's plan-view image representation. This generally causes some true background pixels to be incorrectly labeled with positive feedback, but, as discussed in section 3.1, it is generally harmless when feedback maps are imprecise in this way. The overall positive feedback map is produced by summing the maps generated for the individual people. Negative feedback (with a value of -1) is generated at all foreground pixels not inside any of the individual bounding boxes.

Note that other person detection and tracking methods, including ones that make no use of depth, can be substituted here, provided that the methods label the approximate image locations of people in the scene with a reasonable degree of reliability. In general, it is preferable that the person tracking methods produce false positives (image locations incorrectly labeled as people) rather than false negatives (person detection failures), since repeated failures to detect a person at a particular image location may produce strong negative feedback that causes the person's appearance to be quickly incorporated into the background model. Repeated misclassification of an image region as a person, on the other hand, merely slows adaptation of the background model to changes in that region.

Our second module detects rapid changes in global illumination, camera gain, or camera position. When it detects any of these events, it produces negative feedback (with a value of -1) at all current foreground locations so that the TAPPMOG will quickly update itself to reflect the new scene appearance. The feedback is generated not just at the event onset, but for a time window long enough to allow for good TAPPMOG adaptation to the changes. The module decides that one of these events may have occurred whenever the TAPPMOG suddenly produces a large amount of foreground that is well-distributed about the image. Shape information, from the depth data, is then used to discriminate these events from the possibility that a foreground object has closely approached the camera, occupying most of its field of view. In this case, we do not want to rapidly update the background model.

The final feedback bitmaps $\mathcal{P}$ and $\mathcal{N}$ are produced by thresholding the summed feedback from the two modules at +1 and -1, respectively.

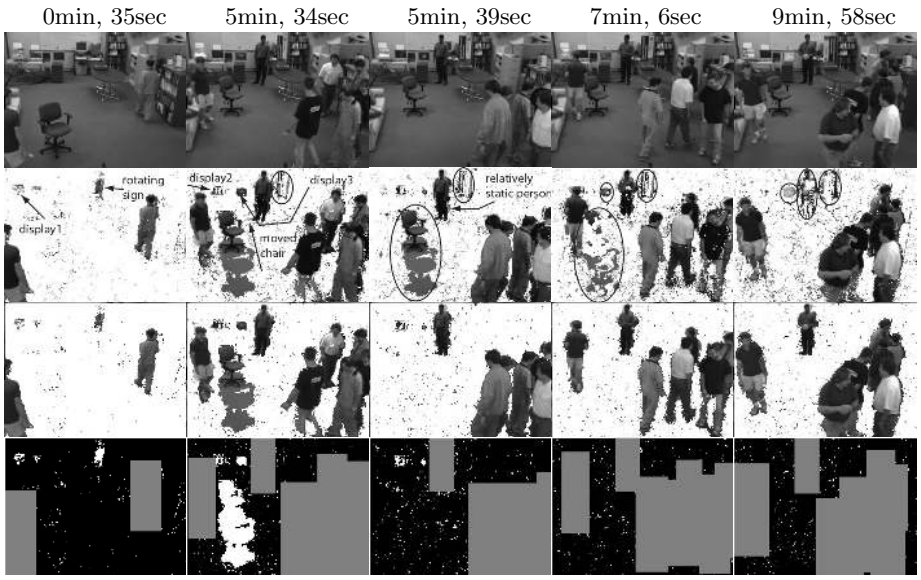


**Fig. 3.** Pixel-level segmentation results with **(third row)** and without **(second row)** high-level feedback correction. Some significant differences are circled in second row; second row also contains some object labels. Input color (**top row**) and depth (not shown) were captured using the Tyzx real-time stereo camera head [18]. Feedback maps (**bottom row**) show $\mathcal{N}$ in white, $\mathcal{P}$ in gray, and "no feedback" in black. TAPPMOG parameters are the same as for Figure 1, but no cleanup of isolated foreground pixels or holes has been done. Feedback parameters: $\kappa = 0.3, \alpha_e = 0.1, \tau = 0.95, \theta = 1.0$, minweightratio = 0.1.

In Figure 3, we compare foreground segmentation results with and without this feedback for the same Tyzx test sequence described in Section 2.3. Several important differences are evident. First, without feedback, the relatively static person toward the back of the room (upper-middle of images) begins to fade into the background after less than 90 seconds of standing at his position (see figure column 4). After a couple more minutes, he becomes difficult to separate from noise, so that tracking and any other analysis of this person becomes very challenging. However, when feedback from the person-tracker is used to prevent background model update at his location, he is well-segmented as foreground for the entire five minutes that he stands at his position, and would remain so indefinitely if the test sequence were longer. This is achieved without sacrificing model adaptation to other scene changes such as the moving of the chair near the midpoint of the sequence. The moved chair causes temporary foreground errors for both methods (see figure column 2). With feedback, these errors are corrected in about 2 seconds, without disrupting segmentation elsewhere. Without feedback, chair errors linger for nearly two minutes, during which the person-tracker must detect and ignore them.

Inclusion of dynamic foreground objects such as the video displays and the rotating sign was virtually eliminated by using negative feedback. Early in the sequence, negative feedback is generated when the sign and displays occasionally

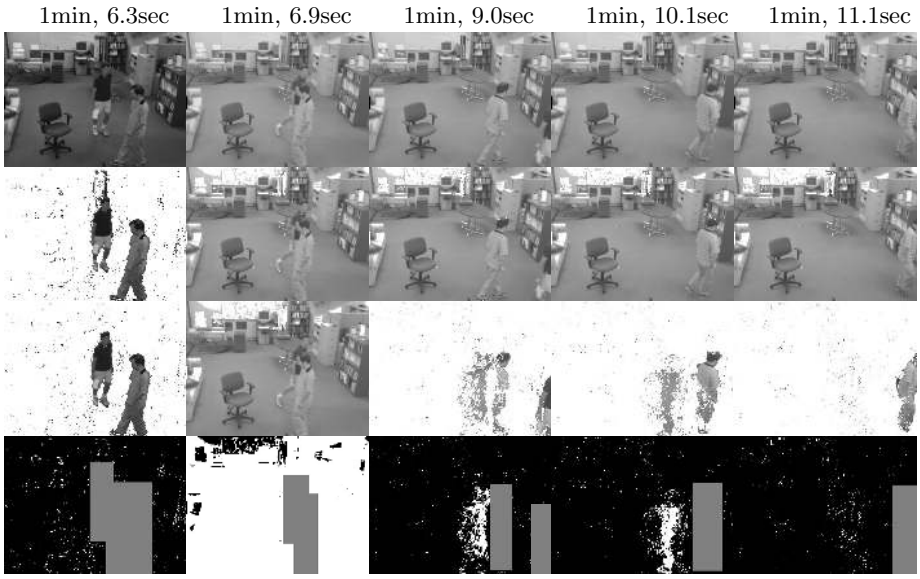


**Fig. 4.** Pixel-level segmentation results with **(third row)** and without **(second row)** high-level feedback correction, for a simulated rapid illumination change. Input color (**top row**) and depth (not shown) were captured using the Tyzx real-time stereo camera [18]. Feedback maps (**bottom row**) show $\mathcal{N}$ in white, $\mathcal{P}$ in gray, and "no feedback" in black. TAPPMOG parameters are the same as for Figure 1, and feedback parameters are the same as for Figure 3. No cleanup of isolated foreground or holes has been done.

appear in the foreground (figure column 1), until they stop appearing. When connected-components analysis is used to extract "significant" foreground objects, the rotating sign appears in no frames beyond the 2-minute mark of the sequence. In contrast, without feedback, the sign appears in 74% of foreground frames beyond the 2-minute mark, including all frames in Figure 3. Similarly, in both sets of results, "display3" sometimes appears in the foreground soon after it is turned on around the 5.5-minute mark. After 2 minutes pass, significant parts of "display3" appear in only 5% of the foreground frames for the feedback-aided method, in contrast to 68% without feedback (figure columns 4 and 5).

The foreground noise levels for the two methods are noticeably different toward the later frames of the test sequence. Without feedback, the variances in the TAPPMOG model drop to very low levels over time, so that imager noise frequently results in color or depth measurements that exceed the tolerance for matching to the background model. With feedback, the system learns to increase these variances where they become problematically small. The lower noise levels result in more cleanly-segmented foreground regions, and less higher-level processing and analysis dedicated to the removal of image clutter.

Segmentation results with and without feedback are compared for a simulated change in global illumination in Figure 4. The lighting change was simulated by applying a gamma correction of 2.0 to a one-minute portion of the test sequence of Section 2.3, beginning at the 1000th frame (a little over a minute into the

sequence). For both methods, almost all of the image appears as foreground immediately after the illumination change (see figure column 2). Without feedback, this condition persists for over 30 seconds while the background model adapts. The adaptation time could be reduced by increasing the TAPPMOG learning rate, but this would further lessen the time that a person, such as the one who appears near the back of the room later in the sequence, can remain relatively still before becoming part of the background. In contrast, when feedback is used, the illumination change is detected and causes negative feedback maps covering most of the frame to be generated for a short period of time. One such map is shown in figure column 2; note that the negative feedback is canceled where the person-tracker module estimates that people are present. Within 2 seconds, the background model is almost completely repaired, except where people occluded the background during the correction period (see figure column 3). When these regions are disoccluded, the person tracker module identifies them as not being people, and generates further, more localized negative feedback that repairs the background model here over the next 2 seconds (see figure columns 3-5).

It is important to note that the correction of most of the background model in under 2 seconds is fast enough to allow person tracking methods to continue to operate through rapid, frame-wide changes. By using prior history and probabilistic methods to estimate person locations during a brief interruption of reliable measurements, tracking can recover gracefully when the background is repaired quickly. Obviously, continuity of tracking is much less likely when an interruption lasts for more than 30 seconds, as was the case without feedback.

## 5 Conclusions

Although TAPPMOG modeling is one of the leading choices for general, robust pixel-level foreground segmentation, we have shown that this robustness can be greatly improved, and tailored to the needs of a specific application, by applying a general framework for guiding the TAPPMOG process with corrective feedback from high-level system modules. Rather than seek to assess high-level semantics using low-level statistics, this framework encourages each module to focus on tasks appropriate to it. The pooling of feedback influences from different high-level modules into a common TAPPMOG also creates a mechanism for high-level modules to share knowledge of errors they detect so that all modules that make use of the pixel-level segmentation may benefit. Finally, feedback can provide a means for fast background adaptation in response to selected events, thereby making it more feasible to slow the learning rate $\alpha$ for the basic TAPPMOG modeling process. The basic TAPPMOG process would therefore better reflect longer-term, persistent features of the scene, and would be less susceptible to influence from foreground objects that remain static too long.

Because the most compute-intensive part of the feedback framework - namely, the error-modeling component - is itself based on TAPPMOG principles, any effort spent on speeding up TAPPMOG background modeling implementations (e.g. in MMX or hardware) can also be leveraged to make feedback correction a relatively light-weight addition. Also, the similar processes for background and error modeling at the pixel level, the guiding of these processes by feedback map

connections from high-level modules, and the resultant focusing of higher-level attention on the foreground of greatest interest, are all reminiscent of aspects of biological visual cortex. We hope to further explore these similarities.

## Acknowledgments

The author would like to thank Tyzx Inc. and Interval Research Corp. for their support in publishing this work and their permission to use data sets obtained at Interval.

## References

1. D. Beymer. "Person Counting using Stereo". In *Wkshp. on Human Motion*, 2000.
2. K. Bhat, M. Saptharishi, P. Khosla. "Motion Detection and Segmentation Using Image Mosaics". In *IEEE Intl. Conf. on Multimedia and Expo 2000*, Aug. 2000.
3. T. Darrell, D. Demirdjian, N. Checka, P. Felzenszwalb. "Plan-view Trajectory Estimation with Dense Stereo Background Models". In *ICCV'01*, July 2001.
4. A. Elgammal, D. Harwood, L. Davis. "Non-Parametric Model for Background Subtraction". In *ICCV Frame-rate Workshop*, Sep 1999.
5. N. Friedman, S. Russell. "Image Segmentation in Video Sequences: a Probabilistic Approach". In *13th Conf. on Uncertainty in Artificial Intelligence*, August 1997.
6. X. Gao, T. Boult, F. Coetzee, V. Ramesh. "Error Analysis of Background Adaptation". In *CVPR'00*, June 2000.
7. M. Harville. "Stereo Person Tracking with Adaptive Plan-View Statistical Templates". HP Labs Technical Report, April 2002.
8. M. Harville, G. Gordon, J. Woodfill. "Foreground Segmentation Using Adaptive Mixture Models in Color and Depth". In *Proc. of IEEE Workshop on Detection and Recognition of Events in Video*, July 2001.
9. K. Konolige. "Small Vision Systems: Hardware and Implementation". *8th Int. Symp. on Robotics Research*, 1997.
10. A. Mittal, D. Huttenlocher. "Scene Modeling for Wide Area Surveillance and Image Synthesis". In *CVPR'00*, June 2000.
11. H. Nakai. "Non-Parameterized Bayes Decision Method for Moving Object Detection". In *ACCV'95*, 1995.
12. J. Ng, S. Gong. "Learning Pixel-Wise Signal Energy for Understanding Semantics". In *Proc. Brit. Mach. Vis. Conf.*, Sep. 2001.
13. J. Orwell, P. Remagnino, G. Jones. "From Connected Components to Object Sequences". In *Wkshp. on Perf. Evaluation of Tracking and Surveillance*, April 2000.
14. Point Grey Research, http://www.ptgrey.com
15. S. Rowe, A. Blake. "Statistical Background Modeling for Tracking with a Virtual Camera". In *Proc. Brit. Mach. Vis. Conf.*, 1995.
16. C. Stauffer, W.E.L. Grimson. "Adaptive Background Mixture Models for Real-Time Tracking". In *CVPR'99*, Vol.2, pp.246-252, June 1999.
17. K. Toyama, J. Krumm, B. Brumitt, B. Meyers. "Wallflower: Principles and Practice of Background Maintenance". In *ICCV'99*, pp.255-261, Sept 1999.
18. Tyzx Inc. "Real-time Stereo Vision for Real-world Object Tracking". Tyzx White Paper, April 2000. http://www.tyzx.com
19. J. Woodfill, B. Von Herzen. "Real-Time Stereo Vision on the PARTS Reconfigurable Computer". *Symposium on Field-Programmable Custom Computing Machines*, April 1997.
20. M. Xu, T. Ellis. "Illumination-Invariant Motion Detection Using Colour Mixture Models". In *Proc. Brit. Mach. Vis. Conf.*, Sep. 2001.
21. R. Zabih, J. Woodfill. "Non-parametric Local Transforms for Computing Visual Correspondence". In *ECCV'94*, 1994.