ADAPTIVE VIDEO BACKGROUND MODELING USING COLOR AND DEPTH

Michael Harville
Hewlett-Packard Labs
1501 Page Mill Rd., Palo Alto, CA 94304
michael.harville@hp.com

Gaile Gordon, John Woodfill
Tyzx Inc.
301 Bryant St., Palo Alto, CA 94301
gaile,john@tyzx.com

ABSTRACT

A new algorithm for background estimation and removal in video sequences obtained with stereo cameras is presented. Per-pixel Gaussian mixtures are used to model recent scene observations in the combined space of depth and luminance-invariant color. These mixture models adapt over time, and are used to build a new model of the background at each time step. This combination in itself is novel, but we also introduce the idea of modulating the learning rate of the background model according to the scene activity level on a per-pixel basis, so that dynamic foreground objects are incorporated into the background more slowly than are static scene changes. Our results show much greater robustness than prior state-of-the-art methods to challenging phenomena such as video displays, non-static background objects, areas of high foreground traffic, and similar color of foreground and background. Our method is also well-suited for use in real-time systems.

1. INTRODUCTION

Many computer vision and image processing systems, particularly those in areas such as surveillance, human-computer interaction, and 3D model reconstruction, rely heavily on an early step, commonly called "background subtraction", that segments the scene into novel ("foreground") and non-novel ("background") regions. While the succeeding analysis steps in these systems are often regarded as more interesting, the systems' overall reliability usually depends as much, if not more, on the accuracy and robustness of their background subtraction methods.

Despite its importance, background subtraction remains a poorly solved problem. The simplest class of methods, often found in experimental systems, use color or intensity as input, and employ a background model that represents the expected background feature values at each pixel as an independent (in image space), static (in time), unimodal (i.e. single-peaked, in feature space) distribution. The model is built during a "learning" phase while the scene is known to be empty of foreground objects, and is never modified thereafter. Typically, foreground is detected on a per-pixel basis wherever the current input image differs significantly from the distribution of expected background values. Although such methods are well-suited for speedy implementation and run time, they fail when confronted with any of a number of common, real-world phenomena, such as changes in the illumination of a scene, shadows and inter-reflections, occasional changes to the true background (e.g. someone removing an object from the scene), foreground object "camouflage" (where the color matches that of the background), high-traffic areas where the true background is usually occluded, and dynamic backgrounds (e.g. video displays, trees swaying in the wind, or rotating fans).

Many methods have attempted to address some, but not all, of these problems, by improving one or more aspects of the basic class of methods. If the background model is allowed to adapt slowly over time and if a luminance-invariant color space is used [7], the system no longer needs to initialize on an empty scene, and gains some robustness to shadows, inter-reflections, gradual illumination changes, and occasional scene modifications. By changing the per-pixel models of expected background features from unimodal probability densities such as single Gaussians, to more sophisticated ones such as Gaussian mixture models [5], one can begin to address complex, non-static backgrounds.

Others have taken advantage of the recent development of real-time depth computation from stereo cameras [3, 4, 6]. Because the shapes of scene objects are not affected by lighting changes, shadows, or inter-reflections, depth data provides much robustness to such phenomena. Depth data is unreliable, however, at scene locations with little visual texture or that are visible to only one camera, and tends to be noisy in general. Hence, background subtraction methods based on depth alone [1] produce unreliable answers in substantial parts of the scene, and often fail to find foreground objects in close proximity to the background, such as feet on a floor. A method using depth and color has been proposed recently [2], but it lacks time adaptivity and uses relatively simple statistical models of the background.

We propose a background subtraction method that, for the first time, combines the best of all of the above ideas. Specifically, it applies dynamically adapting, per-pixel, Gaussian mixture models of expected background appearance to the combined image feature space of depth and luminance-normalized color. The resulting robustness is significantly greater than that of prior algorithms, and yet, provided that real-time depth imagery is available[1], our method can still run in real-time on standard hardware.

2. METHOD

2.1. On-line, per-pixel clustering of observations

The input to the algorithm is a time series of spatially-registered, time-synchronized pairs of color and depth images obtained by static cameras. Each pair of corresponding pixels in the two images provides one scene observation in the combined input space of color and depth. We represent color in the YUV space, which allows us to separate luminance and chroma. The observation at pixel i at time t can be written as $\vec{X}_{i,t} = [\ Y_{i,t} \quad U_{i,t} \quad V_{i,t} \quad D_{i,t}\ ]$.

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

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

[1] Usage of hardware and software for real-time dense depth computation is growing quickly as costs decline rapidly. At least one vendor plans to provide this functionality in the near future at prices around US$100.

where η 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 $\vec{\mu}_{i,t-1,k}$ 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. For notational simplicity, we will denote the kth Gaussian distribution of a mixture as η_k.

To reduce the number of model parameters that must be computed and stored, we assume that all observation component measurements are independent, and that the chroma components have the same variance. This permits us to use diagonal covariance matrices of the form $\Sigma = \mathrm{diag}[\ \sigma_Y^2 \quad \sigma_C^2 \quad \sigma_C^2 \quad \sigma_D^2\ ]$, where σ_Y^2, σ_C^2, and σ_D^2 are the luminance, chroma, and depth variances.

To update a pixel's mixture model as new observations are obtained over time, we would prefer to re-estimate the mixture parameters by an exact Expectation-Maximization procedure each time we add an observation to the pixel's history. Because this would be very computationally expensive, however, we instead use an on-line K-means approximation similar to that of [5]. In this approach, when a new observation $\vec{X}_{i,t}$ at a given pixel is received, we attempt to match it with one of the Gaussians η_k for that pixel. If a match is found, we adapt the parameters of η_k using the current observation; if not, we replace one of the existing η_k with a new one that represents the current observation.

The matching process is carried out by first sorting the η_k in a mixture in order of decreasing weight/variance, and then selecting as a match the first Gaussian whose mean is sufficiently near $\vec{X}_{i,t}$. The sorting causes us to favor matching to η_k that are supported by a large number of previous, consistent observations. A match between $\vec{X}_{i,t}$ and a Gaussian η_k is allowed if each squared difference between corresponding components of $\vec{X}_{i,t}$ and the mean $\vec{\mu}_{i,t-1,k}$ of η_k is less than some small multiple β of the corresponding η_k component variance. The parameter β is typically chosen to be about 2.5 to 3.

We modify this basic matching method, however, to account for the possibility of unreliable chroma or depth data. At low luminance, the chroma components (U and V) of our color representation become unstable. Therefore, when the luminance component of either $\vec{X}_{i,t}$ or the mean of η_k is below some threshold Y_{min}, we do not use the chroma components of either in the matching process. Similarly, because stereo depth computation relies on finding small area correspondences between image pairs, it does not produce reliable depth in regions of little visual texture and in regions, often near depth discontinuities in the scene, that are visible in only one image. Most stereo depth implementations attempt to detect such cases and label them with one or more special values, which we denote collectively as *invalid*. When $\vec{X}_{i,t}$ contains *invalid* depth, or when the fraction of observations modeled by a Gaussian that have *invalid* depth rises above some threshold ρ (we typically set ρ to a relatively low value such as 0.2), we do not use depth in the matching process. Finally, because of the possibilities of unreliable chroma or depth data, we use only the luminance variance, rather than the magnitude of the full covariance matrix, in the sorting of the η_k by their weight/variance ratios.

We further modify the basic matching method by increasing the color matching tolerance β, typically by a factor of 2, when the current depth observation and the depth mean of η_k are both reliable and are a match. This form of "depth-based adaptive color matching", introduced in [2], helps mitigate erroneous foreground inclusions of strong shadows (which match the background in depth but not as well in color) and dynamic background objects, such as video displays or rustling foliage, whose depth remains somewhat constant but whose color at a given pixel is highly variable.

If $\vec{X}_{i,t}$ and some η_k are found to match, we incrementally adapt the parameters of η_k toward $\vec{X}_{i,t}$ as follows:

$$\begin{aligned}
\vec{\mu}_{i,t,k} &= (1-\alpha)\vec{\mu}_{i,t-1,k} + \alpha\vec{X}_{i,t} \\
\sigma^2_{Y,i,t,k} &= (1-\alpha)\sigma^2_{Y,i,t-1,k} + \alpha(Y_{i,t} - \mu_{Y,i,t-1,k})^2 \\
\sigma^2_{C,i,t,k} &= (1-\alpha)\sigma^2_{C,i,t-1,k} + \\
&\quad \alpha(\vec{X}_{C,i,t} - \vec{\mu}_{C,i,t-1,k})^T(\vec{X}_{C,i,t} - \vec{\mu}_{C,i,t-1,k}) \\
\sigma^2_{D,i,t,k} &= (1-\alpha)\sigma^2_{D,i,t-1,k} + \alpha(D_{i,t} - \mu_{D,i,t-1,k})^2
\end{aligned} \tag{2}$$

$\vec{X}_C$ and $\vec{\mu}_C$ denote the chroma subvectors of the pixel observation and the mean of η_k. The parameter α can be interepreted as a learning rate: as α is made smaller, the parameters of η_k will be perturbed toward new observations in smaller incremental steps. Also, $1/\alpha$ effectively determines the duration of the time window of "recent" observations that the Gaussian mixture models represent, with values further in the past being weighted in an exponentially decreasing manner.

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 η_k that matched the observation and 0 (false) for all others, so (3) causes the weight of the matched η_k to increase and all other weights to decay.

If no match is found, the Gaussian ranked last in weight/variance is replaced by a new one with a mean equal to $\vec{X}_{i,t}$, an initially high variance, and a low initial weight (α).

2.2. Background model estimation, 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. We designate the current observation at a pixel to be part of the foreground if it was not matched to any of the η_k in the pixel's current background model.

We select background Gaussians 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 the threshold ρ) and whose normalized weight $w'_k = w_k / \sum_k w_k$ exceeds a low threshold T_D, we select the η_k with the largest depth mean. This criterion is based on the fact that, in general, we do not expect to be able to see through the background. The threshold T_D discourages the selection of a background model η_k based on spurious or transient observations. We set T_D around 0.1 to 0.2, so that we can select an η_k representing the true background even when it is usually not visible.

Next, we select additional Gaussians, in order of decreasing weight/variance, until the total weight of the selected Gaussians exceeds a second threshold T. This is most useful where the true background corresponds to a Gaussian with unreliable depth statistics (because the depth measurements are often labeled *invalid*), or where the background is truly multimodal, such as for swaying trees or a fan blade rotating in front of a wall.

2.3. Activity-based learning rate modulation

As the background learning rate α is increased, static changes to the background, such as the moving of a chair, are correctly incorporated into the background model more quickly. Unfortunately, this also causes the model of the true background to be lost more rapidly in regions of high foreground traffic, and causes true foreground objects that remain relatively static, such as two people

who have stopped to have a conversation, to fade more quickly into the background. We mitigate this tradeoff by exploiting the tendency for the scene "activity" level to be higher in the latter two cases than in the first. Specifically, we compute a scene activity measure A at each pixel, and we reduce the learning rate α (used in the update equations (2) and (3)) by some factor ξ at all pixels where A exceeds a threshold H. A moved, but now static, chair will then continue to be incorporated into the background model quickly, while foreground objects that are not completely motionless will be incorporated less quickly.

The activity level A at each pixel is set to zero for the first time step, and thereafter is computed recursively from the previous activity level and the luminance change since the previous frame:

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

The activity learning rate λ helps temporally smooth the frame differences, and helps to keep the activity level high within the interior of low-texture objects as they move across the scene. The threshold H is set as low as possible without causing imager noise to drive A above H at a substantial fraction of pixels. We typically set ξ in the range of 5 to 20; the higher the value of ξ, the longer non-static foreground objects may obscure the background before being incorporated into it. α is not reduced to zero, so that our method is still able to learn models of dynamic backgrounds.

2.4. Approximations for processing speedup

A number of approximations to our method may be made to significantly increase processing speed with minimal degradation in segmentation accuracy. First, the squared differences in equation (2) may be replaced with absolute differences. The process of matching observations to Gaussians then operates in a linear, rather than quadratic, space of differences. This eliminates many multiplication steps, and was found to cause negligible changes in segmentation results. Next, we found that maintaining each pixel's Gaussian mixture components in order of weight alone, rather than by weight/variance ratio, saved many costly division operations without significantly changing segmentation results in most cases.

Much more substantial speedup can be achieved, however, by updating the background model only periodically, rather for each pixel for every observation frame. When updating a pixel every Nth frame, we adapt the pixel's Gaussian mixture via equations (2) and (3) using the most recent observation, with the learning rate increased to approximate the effect of having updated the model N times with this same observation at the normal learning rate. The mixture components representing background are then re-selected, and this background model is used for segmentation until the next model update. When periodic background updating is used, the calculation of per-pixel activity may also be done periodically, since activity is used only to adjust the learning rate, which in turn is needed only for model updating. In practice, we typically choose an update frequency around 0.25-1Hz. For a system running at 8Hz, a model update frequency of 0.5Hz would mean that the pixel model's update occurs only once every 16 frames. All pixels can be updated during the same frame, but to prevent fluctuations in system frame rate, it is preferable to partition them into equal-sized groups, one of which is updated on each frame. It is better still that each group be spatially distributed (e.g. in a regular grid pattern) to avoid certain segmentation artifacts. Model update, background Gaussian selection, and activity computation represent a substantial fraction of system computation, so performing these operations for only a small fraction of frames at each pixel reduces the method's run time to be near that of static background modeling.

3. EXPERIMENTAL RESULTS

To evaluate our algorithm (hereafter denoted as "method (A)"), we compared its performance, for a challenging video test sequence, to that of several alternative methods, each of which lacks a different single key component of (A). Specifically, we tried (B) removing depth information, (C) removing color information, (D) decreasing the per-pixel background model complexity from mixtures of Gaussians to single Gaussians, and (E) removing activity-based learning modulation.

Our test sequence was obtained by saving the output of a stereo camera pair to files at 320x240 resolution and 15 frames per second, and computing depth offline by a method based on [8]. The sequence is 10 minutes long, with no image being 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 midpoint of the sequence, the chair in the lower left of the image is moved to new floor position, "display2" is switched off, "display3" is switched on, and several more people enter the scene. One of these people stands near the middle of the back of the room, and remains there for the rest of the sequence, occasionally shifting his weight or moving his arms. The other new people walk around continuously in the lower right portion of the image, creating a "high-traffic" area.

For this sequence, the ideal foreground segmentation would, in all frames, contain nothing but the people and any objects they manipulate. The video displays and the rotating sign would always be modeled as part of the background. Foreground objects would be segmented without "holes" (missing pieces) and without their shadows. However, for any pixel-level segmentation method with no input from higher level modules, we also expect some temporary errors while the method adapts to changes made to the true background. For instance, when the chair is moved, we expect to see, for some amount of time, two erroneous foreground regions corresponding to the new and old locations of the chair. We would like to minimize the duration of such errors.

Our experiments indicated that our method came much closer to achieving this ideal segmentation than any of the alternatives. Results for selected illustrative frames are shown in Figure 1. We observe fewer holes in (A) due to color or depth camouflage than in (B) and (C), since (A) has an additional measurement source to disambiguate each of these situations and to build statistically cleaner models of the background. Also, our method rarely included the video displays in the foreground, and usually did not include the rotating sign. This results partly from the presence of depth information, which allows for the building of better models of objects such as these whose color is much more variable than their depth from the camera. Without depth data, method (B) is less able to ignore these objects, as can be seen in the result frames. Also, for the sign in particular, it is evident that the use of multimodal (mixture of Gaussian) background models was important for modeling these objects, since the sign appears in several foreground result frames for the unimodal method (D).

Activity-based learning rate modulation proved very helpful in minimizing temporary errors due to background changes, without compromising robustness to other phenomena. For example, the change in the location of the chair was incorporated into the background model by (A) in less than 2 minutes, while (A) segmented

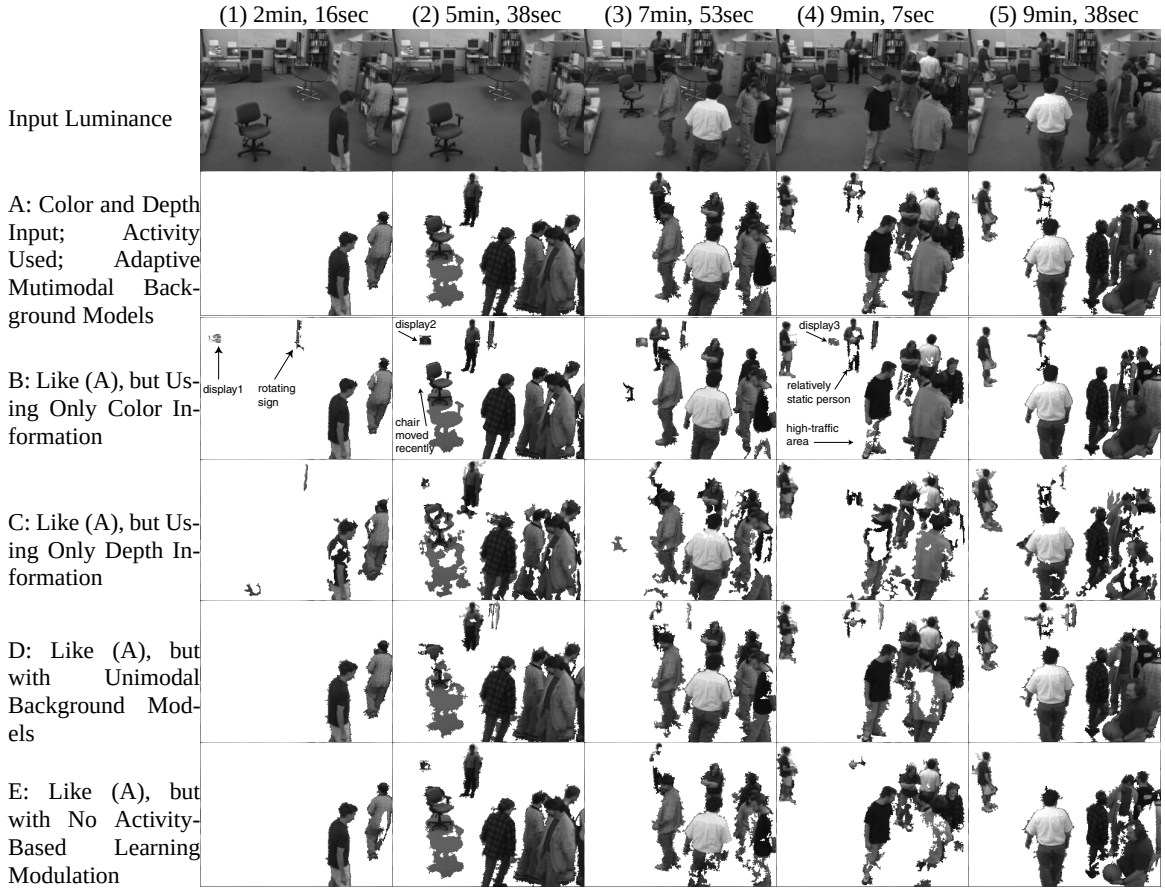

Fig. 1. Comparison of segmentation results for our method to that of close relatives, at selected frames of a challenging video sequence. Several test sequence challenges are indicated in result images for method (B). Where applicable, parameter settings were $K = 4, \alpha = 0.0006, \beta = 2.5, \rho = 0.2, Y_{min} = 16, T = 0.4, T_D = 0.2, \lambda = 0.09, H = 5, \xi = 5$. Small, isolated foreground regions, and small holes within foreground, were removed by applying an area threshold to results of connected-components analysis. Depth input not shown.

people accurately in the high-traffic area for the full 5 minutes of its duration. The slower background model learning rate in the presence of activity largely accounts for this time difference, as can be seen from the more significant foreground omissions in the high-traffic area in the late frame results of (E). This was also the most important factor in enabling (A) to segment a significant portion of the relatively inactive person in the upper-middle of the later frames. Although this person did not move much, he moved enough to cause the learning rate to be lowered at his location much of the time, and he was almost always segmented, at least in part, for the entire 5 that minutes he stood at his position. In contrast, in method (E), he was almost entirely incorporated into the background model after just 1.4 minutes.

For (C), in which color is not available, we improved results by modeling valid and *invalid* depth with separate Gaussians, as in [1], rather than combine them in single Gaussians as described in section 2.1. Despite this, results for (C) were rather poor, largely because of the substantial low-texture image regions, such as the floor, that produce very noisy, often *invalid* depth data.

4. CONCLUSIONS

We have described a new algorithm for background estimation and removal in video that possesses much greater robustness to a variety of common, problematic, real-world phenomena. We demonstrate results for a challenging file sequence, but we are also able to run the system with live color and depth video input. With little attempt at code optimization, the system runs at 8Hz on 320x240 images on a 500MHz Pentium processor. This combination of speed and reliability creates a solid background subtraction platform on which to build a wide variety of computer vision applications for practical, relatively unconstrained settings.

5. REFERENCES

[1] C. Eveland, K. Konolige, R. Bolles, "Background Modeling for segmentation of video-rate stereo sequences", *Proc. CVPR'98*, 1998.
[2] G. Gordon, T. Darrell, M. Harville, J. Woodfill, "Background Estimation and Removal Based on Range and Color", *Proc. CVPR'99*, 1999.
[3] K. Konolige, Small Vision Systems: Hardware and Implementation, *Eighth Intl. Symposium on Robotics Research*, (Hayama, Japan) 1997.
[4] Point Grey Research, http://www.ptgrey.com.
[5] C. Stauffer, W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking", *Proc. CVPR'99*, June 1999.
[6] J. Woodfill, B. Von Herzen, "Real-Time Stereo Vision on the PARTS Reconfigurable Computer", *IEEE Symposium on Field-Programmable Custom Computing Machines*, (Napa, CA) April 1997.
[7] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, "Pfinder: Real-time Tracking of the Human Body", *IEEE Trans. on Pat. Anal. and Mach. Intell.*, 19:7, July 1997.
[8] R. Zabih, J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence", *Proc. ECCV'94*, 1994.