Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter

Zhong, Jing

Boston University Computer Science Department

Boston University Computer Science Tech. Report No. 2003-018, July 18, 2003. Appears in Proc. ICCV 2003.

Segmenting Foreground Objects from a Dynamic Textured Background via a Robust Kalman Filter

Jing Zhong and Stan Sclaroff

Computer Science Department
Boston University, Boston, MA 02215
Email:{jingzh,sclaroff}@cs.bu.edu

Abstract

The algorithm presented in this paper aims to segment the foreground objects in video (e.g., people) given time-varying, textured backgrounds. Examples of time-varying backgrounds include waves on water, clouds moving, trees waving in the wind, automobile traffic, moving crowds, escalators, etc. We have developed a novel foreground-background segmentation algorithm that explicitly accounts for the non-stationary nature and clutter-like appearance of many dynamic textures. The dynamic texture is modeled by an Autoregressive Moving Average Model (ARMA). A robust Kalman filter algorithm iteratively estimates the intrinsic appearance of the dynamic texture, as well as the regions of the foreground objects. Preliminary experiments with this method have demonstrated promising results.

1 Introduction

This paper describes a framework for detection and segmentation of foreground objects in video, when the background is a dynamic texture. In a broad sense, dynamic textures exhibit repetitive patterns in space-time. Example dynamic textures are shown in Fig. 1. Foreground objects are those that appear in front of a dynamic texture with distinctive statistics in space-time. Real world examples include ships on the sea, people riding an escalator, etc. It is assumed that foreground objects are distinctive in at least their spatial *or* temporal statistics. For instance, the foreground objects and dynamic, textured background might have similar color distributions, but differ in their motion patterns.

The non-stationary nature and clutter-like appearance of dynamic textures cause many traditional background subtraction methods to fail, since these methods assume a static or slowly changing background. We propose an algorithm that explicitly models the dynamic, textured background via an Autoregressive Moving Average (ARMA) model [16]. Although ARMA is a first-order linear model many dynamic textures can be well captured by it [3]. A robust

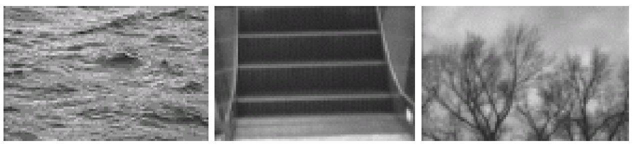

Figure 1: Examples of dynamic textures.

Kalman filter algorithm is used in estimating the intrinsic appearance of the dynamic texture. The foreground object regions are then obtained by thresholding the weighting function used in the robust Kalman filter.

2 Related work

A fundamental problem in surveillance and monitoring is the extraction of the interesting foreground regions or objects, given an incoming video of the scene. Several foreground-background segmentation algorithms have been proposed to solve this problem, including algorithms based on background subtraction, color distributions, motion, as well as range and stereo.

In one of the earliest works [9] the foreground object is detected by subtracting the current picture from the stored background picture, and then applying a threshold to the resulting difference image. The method obviously fails if the background is dynamic. Recently, it has become popular to model the pixel-wise color distribution of the background though statistical methods. In [4, 17] the color distribution of each background pixel is learned using a Gaussian mixture model, and these mixture models are continuously updated to adapt to slow background changes. In [14] each pixel is modeled using a Kalman filter. These methods all address the problem of segmentation given a dynamic background. However, they do not take advantage of inter-pixel correlation and global appearance. Thus, they may fail to extract objects when the color distributions of the foreground and background are similar.

Other approaches such as [6, 11] use region-based features (e.g., block correlation) to detect the foreground object(s). Change detection between consecutive frames is achieved via block matching; thus, the foreground object(s) can only be detected at a coarse scale unless a multiscale-based approach is used.

Some approaches take advantage of depth information. Background subtraction methods that rely only on depth [8] may be unreliable when the foreground objects are too close to the background. Combined use of depth and color is proposed in [5]. However, the approach assumes the background scene is relatively static. In addition, only indoor experimental testing was reported.

Motion-based approaches have also been proposed. For instance [22] proposes an algorithm to detect salient motion by integrating frame-to-frame optical flow over time; thus, it is possible to predict the motion pattern of each pixel. The saliency measure is then computed to detect the object locations. This approach assumes that the object tends to move in a consistent direction over time, and that foreground motion has different saliency. This algorithm may fail when there is no obvious difference between the motion fields of the foreground and background.

Some approaches exploit spatiotemporal intensity variation. For instance [10, 12] employ analysis of XT or YT video slices. By detecting the translational blobs in these slices, it is possible to detect and track walking people and recognize their gait. The method may perform inaccurately if both foreground and background objects exhibit periodic motion, as in the case of some dynamic textures. A related approach [21] assumes that the foreground object is moving with a certain velocity.

Robust statistical methods could help in improving foreground-background segmentation methods. Robust methods have been used in motion estimation [1], and in conic fitting [23]. In [19] a robust Kalman filter is used in tracking explicit curves. In [2] a robust principal components analysis (PCA) formulation is proposed that addresses the intra-sample outlier problem of conventional PCA training. In [13] a robust Kalman filter framework for recovery of moving objects' appearance is proposed; however, the framework does not model dynamic, textured backgrounds – only the dynamic foreground. Furthermore, the method does not account for the state covariance in the Kalman filter's update equations. Related work in robust Kalman filters can be found in Communication and Control; e.g., [20], which considers parameter disturbance, and the robust Kalman filter is realized via convex optimization.

3 Dynamic texture formulation

Following [16] a dynamic texture can be formulated via a generative model. Suppose that at each time instant t we observe a noisy version of the image $I(t) = CX(t)+u+w(t)$, where $w(t) \sim N(0, R)$ is the Gaussian probability density function with covariance R, $u \in \Re^m$ is a constant vector (DC), and m is the number of the pixels in the image. The vector $X \in \Re^n$ describes the dynamic texture's state. The orthogonal matrix C is $m \times n$, where $m >> n$. The columns of this matrix could be principal components, wavelet filter banks, etc. To simplify our derivation, we will assume that the DC component is removed, and therefore our observations take the form $Y(t) = CX(t) + w(t)$. In our notation we will define $X_t \equiv X(t)$, and $Y_t \equiv Y(t)$. Thus, the autoregressive model of dynamic texture is:

$$\begin{array}{rcll} X_{t+1} &=& AX_t + v_t, & X_0 = x_0; v_t \sim N(0,Q) \\ Y_t &=& CX_t + w_t, & w_t \sim N(0,R) \end{array} \quad (1)$$

where the state evolution matrix A is $n \times n$ and assumed stable $|A| < 1$, and the state noise is $v_t \sim N(0, Q)$. The generative model is:

$$\begin{array}{rcl} p(X_{t+1}|X_t) &=& s_1 e^{-\frac{1}{2}(X_t - AX_{t-1})^T Q^{-1}(X_t - AX_{t-1})} \\ p(Y_t|X_t) &=& s_2 e^{-\frac{1}{2}(Y_t - CX_t)^T R^{-1}(Y_t - CX_t)} \end{array} \quad (2)$$

where $s_1 = 1/\sqrt{(2\pi)^n|Q|}$ and $s_2 = 1/\sqrt{(2\pi)^m|R|}$ are used to normalize the exponential form.

Estimation of the model parameters can be realized using PCA and maximum-likelihood training [16]. Robust PCA methods [10] could also be used in training.

4 Object detection and segmentation

The challenge of foreground segmentation given dynamic textured scenes is that the background is continuously changing. This causes classical foreground-background techniques to fail. Alternative methods, e.g., modeling the color distribution for each pixel [4], may fail if the object has a similar color distribution to that of the background.

We propose a novel method that estimates the state parameters of the dynamic, textured background in the presence of the foreground objects. Given an estimate of the dynamic background's appearance, extraction of the foreground objects is easily attained. We formulate this approach using a robust Kalman filter to iteratively update the state of the dynamic texture ARMA model and to determine a mask image for the foreground objects.

4.1 Dynamic background updates

If no foreground objects were present, the new state of the dynamic texture could be estimated by maximizing the probability of X_t given the observation Y_t and previous state sequence $X_t^- = [X_{t-1}^T, X_{t-2}^T, \ldots, X_0^T]^T$ used to predict the current state. To simplify the model, we use the Markov assumption, i.e., the current state only depends on the previous one $X_t^- = X_{t-1}$. Thus

$$\hat{X}_t = \arg\max_X p(X_t|Y_t, X_{t-1}). \tag{3}$$

Using Bayes formula, the posterior probability becomes:

$$p(X_t|Y_t, X_{t-1}) = \frac{p(Y_t|X_t, X_{t-1})p(X_t|X_{t-1})}{p(Y_t)} \tag{4}$$

where $p(Y_t)$ is prior distribution of observation Y_t. The observation Y_t is independent of the previous state X_{t-1} in the presence of X_t, since Y_t is generated by the state X_t only. Thus (3) is reduced to:

$$\hat{X}_t = \arg\max_X p(Y_t|X_t)p(X_t|X_{t-1}). \tag{5}$$

The state inference can be converted to the standard Kalman filter solution if we assume that the observation density and state transition density are Gaussian distributed. The state estimate can then be obtained by minimizing the following criterion function, by taking the logarithm of (5) and (2):

$$\begin{aligned} J &= (Y_t - C\hat{X}_t)^T R^{-1}(Y_t - C\hat{X}_t) + \\ & \quad (\hat{X}_t - \hat{X}_t^-)^T Q^{-1}(\hat{X}_t - \hat{X}_t^-) \end{aligned} \tag{6}$$

where $\hat{X}$ is the estimated and $\hat{X}^-$ the predicted value:

$$\hat{X}_t^- = A\hat{X}_{t-1}^-. \tag{7}$$

4.2 Updates in the presence of outliers

Foreground objects can be considered as outliers of the dynamic texture model. In the above Kalman filter formulation, these outliers would corrupt our estimates of the dynamic texture parameters; as a consequence foreground-background subtraction performance would be significantly degraded. To address this, the state vector estimation problem can be reformulated using robust estimation. Since we assume R is a diagonal matrix, the first term of (6) can be written as:

$$J_1 = \sum_{i=1}^{m} \left((Y_i - [C\hat{X}]_i)/\sigma_i\right)^2, \ \sigma_i = \sqrt{R_i} \tag{8}$$

which corresponds to the weighted least square criterion. For simplicity, the subscript t is omitted here. Robust estimation can be formulated as an *M-estimator* [7] which constructs the function as:

$$J_i = \sum_{i=1}^{m} \rho((Y_i - [C\hat{X}]_i)/\sigma_i) \tag{9}$$

where ρ is the robust error norm. The minimization of (8) can be realized by solving:

$$\begin{gathered} \sum_{i=1}^{m} \psi(z_i)\frac{\partial z_i}{\partial x_j} = 0, \ \text{for } j = 1, \ldots, n \\ z_i = (Y_i - [C\hat{X}]_i)/\sigma_i \end{gathered} \tag{10}$$

Here ψ is the influence function for the robust error norm employed [7]. If we define a weighting function:

$$w(z_i) = \frac{\psi(z_i)}{z_i}. \tag{11}$$

Then (10) becomes:

$$\sum_{i=1}^{m} w(z_i) z_i \frac{\partial z_i}{\partial x_j} = 0, \ \text{for } j = 1, \ldots, n \tag{12}$$

This yields the re-weighted least squares problem:

$$\hat{X}^k = \arg\min_X \sum_{i=1}^{m} w(z_i^{k-1}) z_i^2. \tag{13}$$

Here the function w is evaluated in each step k, given the new state estimates. The form of function w depends on the robust error norm employed. Substituting (13) in (10) we obtain:

$$\sum_{i=1}^{m} w(z_i^{k-1}) z_i \frac{\partial z_i}{\partial x_j} = 0, \ \text{for } j = 1, \ldots, n \tag{14}$$

We can rewrite the above equation in matrix form, and then the criterion function for the robust Kalman filter can be written:

$$\begin{aligned} J^k &= (Y - C\hat{X}^k)^T M^{k-1} R^{-1}(Y - C\hat{X}^k) + \\ & \quad (\hat{X}^k - \hat{X}^-)^T Q^{-1}(\hat{X}^k - \hat{X}^-) \end{aligned} \tag{15}$$

where M^{k-1} is the weighting matrix at step k:

$$M^{k-1} = \mathrm{diag}(w(z_1^{k-1}), w(z_2^{k-1}), \ldots, w(z_m^{k-1})). \tag{16}$$

The matrix M^{k-1} is updated in each step k using the new state $\hat{X}^k$.

In summary, the robust Kalman filter algorithm is:

1. Compute the modified PCA matrix
$$C' = (CP_t^- C^T + R)M^T C(P_t^- C^T M^T C)^{-1} \tag{17}$$
2. Compute Robust Kalman Gain
$$K = (C'^T MC')^{-1} C'^T \tag{18}$$
3. Update estimate with measurement Y_t
$$\hat{X}_t = \hat{X}_t^- + KM(Y_t - C\hat{X}_t^-) \tag{19}$$
4. Update the error covariance P_t
$$P_t = (I - KMC)P_t^- \tag{20}$$

The derivation of these equations is given in the Appendix.

The computational complexity of the above process is $O(m^2 n + n^3)$, where m is the dimension of observation vector, and n is the dimension of ARMA model parameters. The m^2 term dominates, since in our application $m >> n$.

4.3 Segmentation algorithm summary

The algorithm for segmenting foreground objects from dynamic texture scenes can now be summarized:

1. Form the foreground mask image by calculating the weighting function, $w(z_i)$, given the observation vector Y and the initial state vector X_0 (calculated *a priori* in the absence of foreground objects).

2. Use the mask image to form the diagonal weighting matrix M and solve for X via the robust Kalman filter.

3. Go to step 1 until the maximum iteration number is reached (fewer than 5 in our implementation).

In our experiments, we used the Cauchy robust error norm. The weighting function for this norm is

$$w(z_i) = \frac{1}{1 + (z_i/c)^2} \tag{21}$$

As for the selection of the parameter c, in principle, smaller values of c make the detector more sensitive to noise, and can lead to a higher likelihood of false alarms. The parameter c can be selected as a multiple of the average of the dynamic texture state variance σ_i. Alternatively, a appropriate value for c can be determined empirically by examining the ROC curve generated with appropriate test sequences of texture and foreground objects.

5 Experiments

In the experiments, we first reduce the width and height of the input images by half so that the program can be run efficiently. This leads to the observation vectors with dimension about 20000. The number of eigenvectors we used is 80 to keep 95 percent of energy. We implemented the algorithm in Matlab 6.1, and the computation speed is about 8 seconds per frame on an Athlon, 1.6 GHz PC.

Both synthesized and natural data are used for testing our system. Among these data, 96 frames are used to train the parameters for the ARMA model, including state transition matrix, measurement error covariance, state error variance and the initial state vector. In addition, both training data and testing data are grayscale images converted from the original color images such that the algorithm cannot take advantage of the color information.

The synthesized test videos include dynamic texture sequences selected from [18]. The synthesized images are created using these dynamic texture images as background, with moving foreground shapes added. The testing image size is 170 $\times$ 115. The color distribution of the synthetic moving object is similar to that of the background; this enables us so to test the algorithm's ability to detect objects that have similar color distributions with the background. Example images from the experiment are shown in Fig. 2.

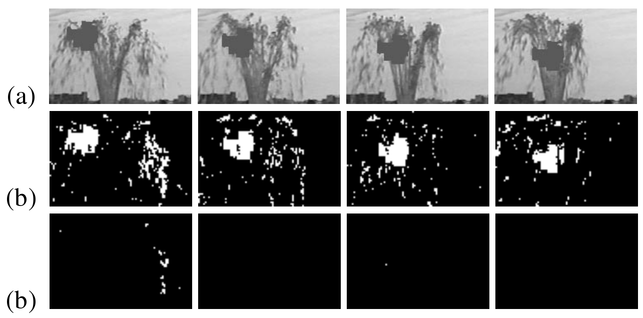

Figure 5: The detection result used GMM as a comparison. (a) Original images sequence. They are the same as the sequence in Fig. 2. (b) Using high threshold. (c) Using low threshold.

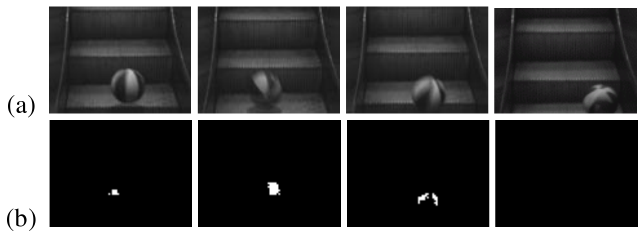

Figure 6: Another example of detection used GMM. (a) Original image sequence (same as the sequence in Fig. 3). (b) The results under the best threshold setting.

Yet another data set consists of video segments captured from natural scenes and objects. The frame size of the video is 320×240. We reduce the widths and heights of the video frames by a factor of 2. The detection results are shown in Figs. 3 and 4.

For comparison, we implemented a simplified Gaussian Mixture Model (GMM) approach. The sophisticated version with parameter updating is described in [4, 17]. The parameter updating is not implemented here. But since we use a short sequence to test, this implementation and results suffice to show the weakness of the GMM-based approach if the foreground and background have similar grayscale distributions, especially if the background has substantial motion. As shown in Fig. 5, the GMM based algorithm cannot accurately detect the objects by either setting a high threshold, or low threshold. In Fig. 6, we deliberately adjust the threshold so that the simulated model can achieve the best result. It is clear that when the grayscale distribution of the foreground and background objects is similar, the GMM-based algorithm fails to extract the foreground objects (the fountain case) or output accurate results (the escalator case).

6 Conclusion

In this paper, we proposed an algorithm that segments foreground objects from dynamic, textured backgrounds. We

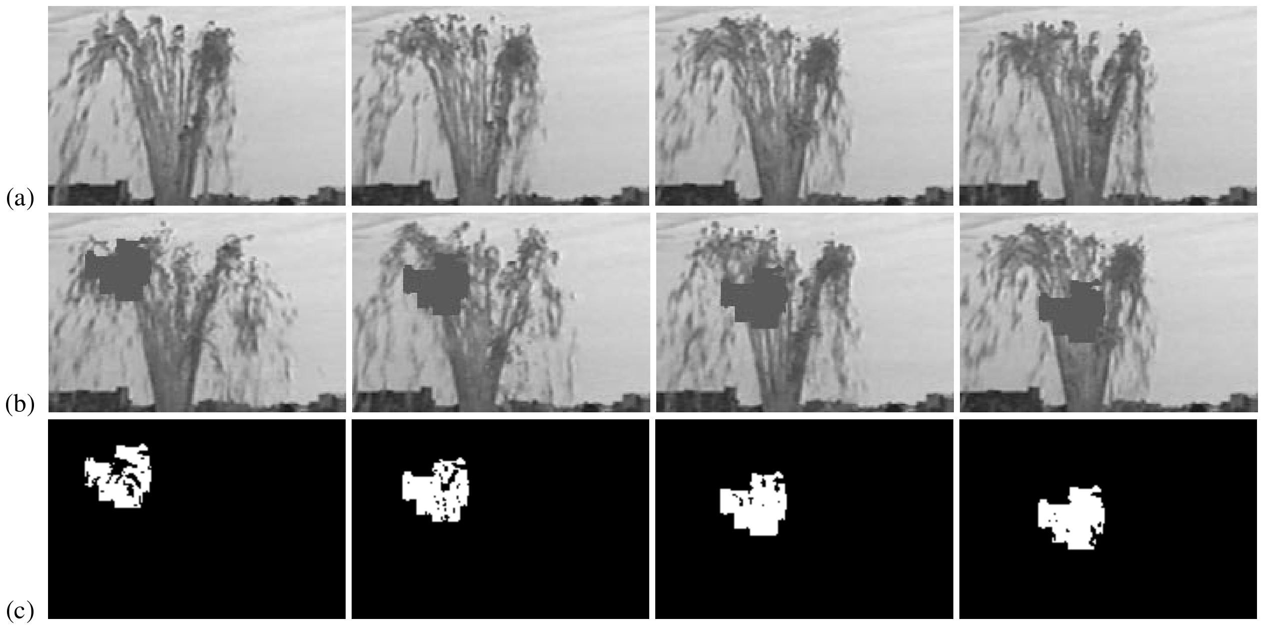

Figure 2: Example of foreground object segmented from dynamic texture scene. The object is synthesized using similar color and with irregular shape. (a) Sequence of images with moving synthesized foreground object. (c) The detected object region.

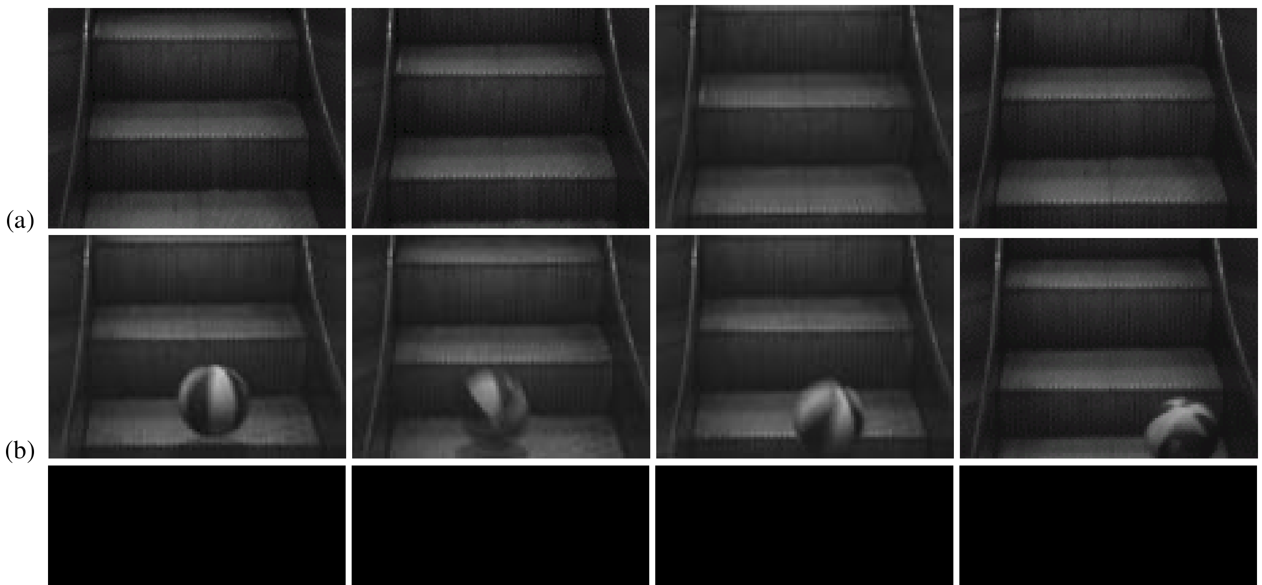

Figure 3: Another example of foreground object detection in real data. (a) Image sequence of moving escalator. (b) Moving escalator with ball. (c) Detected object region.

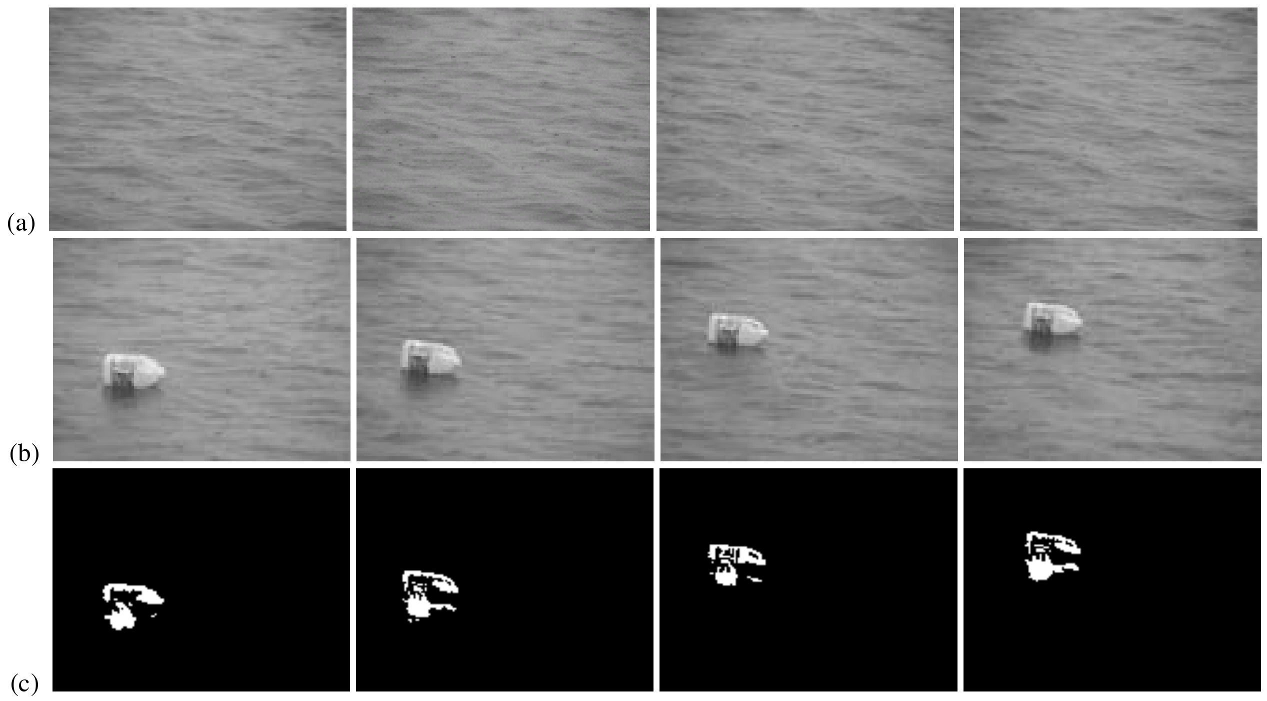

Figure 4: The third example of foreground objects detection in real data. (a) Sequence of waving river. (b) Waving river with floating bottle. (c) Detected foreground object region.

derived robust Kalman filter equations to achieve this purpose. The experiments showed that the algorithm can successfully segment the foreground objects, even if they share a similar grayscale distribution with the background. Furthermore, segmentation results could be further improved via the customary post-processing via morphological operations on the foreground mask.

Many extensions can be made to the basic algorithm. For example, the spatiotemporal consistency of the foreground objects can be modeled to stabilize the robustness of foreground object extraction, using a method such as proposed in [15]. In our current implementation, the Kalman filter model is only trained using the empty scenes; however, we could use the Robust PCA method in [2] to train the ARMA model on non-empty scenes in the future. Furthermore, to account for background changes over time, the parameters could be updated in a certain time period, and some preprocessing could be adopted to filter out noise, for instance, by using band-pass filters. Finally, our current ARMA model only takes grayscale images as input; however, the model could be readily extended to color imagery [3].

References

[1] M.J. Black and P. Anandan. A framework for the robust estimation of optical flow. *Proc. ICCV*, pp. 231–236, 1993.

[2] F. de la Torre and M.J. Black. Robust principal component analysis for computer vision. *Proc. ICCV*, pp. 362–369, 2001.

[3] G. Doretto, A. Chiuso, Y.N. Wu, and S. Soatto. Dynamic textures. *IJCV*, 51(2):91–109, 2003.

[4] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Proc. ECCV*, 2:751–767, 2000.

[5] G. Gordon, T. Darrell, M. Harville, and J.Woodfill. Background estimation and removal based on range and color. *Proc. CVPR*, 2:459–464, 1999.

[6] Y. Hsu, H. Nagel, and G. Rekers. New likelihood test methods for change detection in image sequences. *Computer Vision Applications*, 5:17–34, 1992.

[7] P.J. Huber. *Robust Statistics*. John Wiley and Sons, 1981.

[8] Y. Ivanov, A. Bobick, and J. Liu. Fast lighting independent background subtraction. *IJCV*, 37(2):199–207, 2000.

[9] M.K. Leung and Y.H. Yang. Human body motion segmentation in a complex scene. *Pattern Recognition*, 20:55–64, 1987.

[10] F. Liu and R. Picard. Finding periodicity in space and time. *Proc. ICCV*, pp. 376–383, 1998.

[11] T. Matsuyama, T. Ohya, and H. Habe. Background subtraction for nonstationary scenes. *Proc. ACCV*, pp. 662–667, 2000.

[12] S. Niyogi and E. Adelson. Analyzing and recognizing walking figures in xyt. *Proc. CVPR*, pp. 469–474, 1994.

[13] R.P.N. Rao. Robust Kalman filters for prediction, recognition, and learning. TR 645, U. Rochester, Comp. Sci., 1996.

[14] C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using Kalman-filtering. *Proc. Int. Conf. on Recent Advances in Mechatronics*, pp. 193–199, 1995.

[15] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. *Proc. CVPR Workshop on the Interpretation of Visual Motion*, 1998.

[16] S. Soatto, G. Doretto, and Y.N. Wu. Dynamic Textures. *Proc. ICCV*, pp. 439–446, 2001.

[17] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. *Proc. CVPR*, 2:246–252, 1999.

[18] M. Szummer and R.W. Picard. Temporal texture data set. ftp://whitechapel.media.mit.edu/pub/szummer/temporal-texture/, 1996.

[19] J.P. Tarel, S.S. Ieng, and P. Charbonnier. Using robust estimation algorithms for tracking explicit curves. *Proc. ECCV*, pp. 492–507, 2002.

[20] F. Wang and V. Baladrishnan. Robust Kalman filters for linear time-varying systems with stochastic parametric uncertainties. *IEEE Trans. on Signal Processing*, 50(4), 2002.

[21] R. Wildes. A measure of motion salience for surveillance applications. *Proc. ICIP*, pp. 183–187, 1998.

[22] L. Wixson. Detecting salient motion by accumulating directionally-consistent flow. *PAMI*, 22(8), pp. 774–780, 2000.

[23] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing*, 15:59–76, 1997.

Appendix

Note: In order to have notation consistent with most Kalman Filter derivations, the symbol *H* will be used to denote the state evolution matrix. In our approach *H* = *C*, as described in Sec.4.1

Appendix 1. Derivation of State Estimation

In order to obtain the optimal estimation of the state vector, we need to minimize the following error criterion function:

$$\begin{aligned} J &= (Y - H\hat{X})^T M^k R^{-1}(Y - H\hat{X}) + \\ &\quad (\hat{X} - \hat{X}^-)^T Q^{-1}(\hat{X} - \hat{X}^-) \qquad (1) \\ \frac{\partial J}{\partial \hat{X}} &= -2(Y - H\hat{X})^T (M^k R^{-1} H) \\ &\quad +2(\hat{X} - \hat{X}^-)^T Q^{-1} = 0 \\ &\Rightarrow (Y - H\hat{X})^T (M^k R^{-1} H) = (\hat{X} - \hat{X}^-)^T Q^{-1} \\ &\Rightarrow (Y^T - \hat{X}^{-T} H^T) M^k R^{-1} H \\ &\quad + \hat{X}^{-T}(Q^{-1} + H^T M^k R^{-1} H) \\ &= \hat{X}^T (H^T M^k R^{-1} H + Q^{-1}) \qquad (2) \end{aligned}$$

Let $N = (H^T M^k R^{-1} H + Q^{-1})$ and $K = N^{-1} H^T R^{-1}$. Then

$$\begin{aligned} (2) \quad &\Rightarrow \quad \hat{X}^T = (Y^T - \hat{X}^{-T} H^T) M^k R^{-1} H N^{-1} + \hat{X}^{-T} \\ &\Rightarrow \quad \hat{X} = K M^k (Y - H\hat{X}^-) + \hat{X}^- \qquad (3) \end{aligned}$$

This is the modified state vector update equation in the presence of the diagonal weighting matrix M.

Appendix 2. Derivation of Kalman Gain

According to (3), we seek to minimize the estimation error:

$$\hat{X} = \hat{X}^- + KM(Y - H\hat{X}^-) \qquad (4)$$

The state prediction error vector is $e^- = X - \hat{X}^-$, and the state estimation error vector is $e = X - \hat{X}$. Therefore

$$\begin{aligned} e &= X - \hat{X} \\ &= X - \hat{X}^- - KM(Y - H\hat{X}^-) \\ &= e^- - KM(Y - H\hat{X}^-) \qquad (5) \end{aligned}$$

The overall estimation error is:

$$\begin{aligned} e^2 &= E[e^T e] \\ &= E[(e^- - KM(Y - H\hat{X}^-))^T (e^- - KM(Y - H\hat{X}^-))] \end{aligned} \qquad (6)$$

In order to minimize the estimation error, we set the partial differential of the error respect to K equal to zero

$$\frac{\partial e^2}{\partial K} = -2E[(e^- - KM(Y - H\hat{X}^-))((Y - H\hat{X}^-)^T M^T)] = 0 \qquad (7)$$

Note that $M(Y - H\hat{X}^-) = M(He^- + v)$

$$\begin{aligned} (7) \quad &\Rightarrow \quad E[(e^- - KMHe^- - KMv)((e^{-T} H^T + v^T) M^T)] \\ &= \quad P^- H^T M^T - KMHP^- H^T M^T - KMRM^T = 0 \\ &\Rightarrow \quad K\underline{M(HP^- H^T + R)M^T} = P^- H^T M^T \qquad (8) \end{aligned}$$

Note that the underlined equation is singular. We need to use pseudo-inverse to get the solution:

$$K = (H'^T M H')^{-1} H'^T \qquad (9)$$

where $H' = (HP^- H^T + R) M^T H (P^- H^T M^T H)^{-1}$.

Appendix 3. Derivation of State Covariance

Again we note that $M(Y - H\hat{X}^-) = M(He^- + v)$. Plugging this into (4) we get:

$$e = e^- - KM(He^- + v) = (I - KMH)e^- - KMv \qquad (10)$$

Then the state covariance matrix is:

$$\begin{aligned} P &= E[ee^T] \\ &= (I - KMH)P^-(I - KMH)^T + KMRM^T K^T \\ &= (I - KMH)P^- \\ &\quad + \underline{[(KMH - I)P^- H^T M^T + KMRM^T]} K^T \qquad (11) \end{aligned}$$

The underlined term is zero according to (8), thus

$$P = (I - KMH)P^-. \qquad (12)$$