Segmenting Dynamic Textures with Ising Descriptors, ARX Models and Level Sets

Atiyeh Ghoreyshi and René Vidal

Center for Imaging Science, Department of BME, Johns Hopkins University
308B Clark Hall, 3400 N. Charles St., Baltimore, MD 21218, USA
ati,rvidal@cis.jhu.edu

Abstract. We present a new algorithm for segmenting a scene consisting of multiple moving dynamic textures. We model the spatial statistics of a dynamic texture with a set of second order Ising descriptors whose temporal evolution of is governed by an AutoRegressive eXogenous (ARX) model. Given this model, we cast the dynamic texture segmentation problem in a variational framework in which we minimize the spatial-temporal variance of the stochastic part of the model. This energy functional is shown to depend explicitly on both the appearance and dynamics of the scene. Our framework naturally handles intensity and texture based image segmentation as well as dynamics based video segmentation as particular cases. Several experiments show the applicability of our method to segmenting scenes using only dynamics, only appearance, and both dynamics and appearance.

1 Introduction

A fundamental problem in computer vision is to separate an image into multiple regions of coherent intensity, color or texture. In the case of intensity-based segmentation, several approaches have been proposed over the past few decades. One of the most common methods is based on finding a piecewise smooth approximation of the image by minimizing the Mumford-Shah energy functional [1]. In the case of a piecewise constant approximation, an image $u(x, y)$ is segmented into two regions by finding a curve C of small length $|C|$, a mean intensity c_1 inside C, and a mean intensity c_2 outside C that minimize the energy functional

$$E(C, c_1, c_2) = \mu|C| + \lambda_1 \int_{in(C)} (u(x, y) - c_1)^2 \, dxdy + \lambda_2 \int_{out(C)} (u(x, y) - c_2)^2 \, dxdy. \quad (1)$$

In order to solve this optimization problem, notice that if C were known, then the optimal solution for c_1 and c_2 would be the mean intensities inside and outside C, respectively. Thus, the main challenge in minimizing E is the computation of the optimal C, which requires solving a partial differential equation. This has motivated the development of several methods for efficiently representing C. Explicit methods [2] represent C with a finite number of control points which are evolved to match the boundaries in the scene. Implicit methods [3–5] represent

C as the zero level set of an implicit function φ, i.e. $C = \{(x, y) : \varphi(x, y) = 0\}$, and evolve this function to match the boundaries in the scene.

The main advantages of level set methods over explicit methods are that (1) they do not depend on a specific parametrization of the contour, hence there is no need for re-griding the control points during evolution, and (2) they allow the contour to undergo topological changes such as merging and splitting during evolution. This has motivated various extensions of level set methods from intensity-based image segmentation [4] to texture-based image segmentation [6], motion-based video segmentation [7] and segmentation of *dynamic textures* [8].

Dynamic textures are video sequences of nonrigid scenes whose temporal evolution exhibits certain stationarity, e.g., video sequences of water, fire, smoke, steam, foliage, etc. The works of [9, 10] deal with scenes in which a static camera observes a single dynamic texture. They show that by modeling the temporal evolution of the image intensities as the output of a time invariant autoregressive moving average (ARMA) model, it is possible to jointly recover a model for the appearance and dynamics of the scene using classical system identification techniques [11]. Once these models have been learnt, one can use them to generate novel synthetic sequences [12], manipulate real ones [13], and recognize one from another [14, 9]. The works of [15, 16] extend these methods to scenes containing a dynamic texture observed by a moving camera. [15] introduces the concept of stochastic rigidity which searches for the camera motion that leads to the dynamical model of minimum order. Solving this problem is, however, very computationally intense. [16] models the scene with a time-varying ARMA model from which one can compute the optical flow of the scene using the so-called dynamic texture constancy constraint.

Existing works dealing with multiple dynamic textures include [17, 8, 16, 18]. [17] models the scene as the output of a mixture of ARMA models and learns the parameters of this mixture model and the segmentation of the scene using Expectation Maximization (EM). Unfortunately, EM-like approaches are very sensitive to good initialization. [16] shows that when the sequence is modeled as the output of a mixture of ARMA models, the trajectories of the image intensities live on a mixture of subspaces. The scene is then segmented by clustering these trajectories using GPCA [19]. Unfortunately, GPCA does not incorporate spatial regularization, thus the resulting contour is typically non smooth. [18] incorporates spatial regularization by using spatial-temporal ARX models combined with GPCA. The closest approach to ours is [8], which proposes to segment the scene by minimizing an energy functional using level sets. The energy functional depends on the subspace angles between the observability subspace of a locally computed ARMA model and that of a reference model. This purely algebraic choice of the energy functional is motivated by the fact that the parameters of an ARMA model live on a non-Euclidean space. Therefore, defining and minimizing a statistically sensible energy functional is nontrivial.

In this paper, we conjecture that dynamical models for dynamic texture segmentation need not be as complex as those for synthesis. Therefore, we propose to use simple autoregressive exogenous (ARX) models to describe the temporal

evolution of a set of static texture descriptors. This new dynamic texture model leads to a natural spatial-temporal generalization of the classical Mumford-Shah energy functional for dynamic texture segmentation that has several advantages:

1. First, the parameters of an ARX model live on a Euclidean space, allowing one to define a statistically sensible energy functional that depends on both the appearance and dynamics of the scene.
2. Second, the identification of the parameters of an ARX model can be done in closed form by solving a simple linear system.
3. Third, as we will show experimentally, a good segmentation of the scene can be obtained using ARX models of very low order. In fact, our experiments will show the superiority of our method with respect to existing algebraic and variational approaches which use more complex models of higher orders.
4. Finally, we demonstrate that our method can be easily extended for segmenting dynamic textures with a moving contour.

2 Review of Intensity-Based Image Segmentation

Let $u : \Omega \to \mathbb{R}$ be a given image with domain $\Omega \subset \mathbb{R}^2$. Let $C \subset \Omega$ be a closed contour dividing the image into two regions of coherent intensities. As proposed by [20] one can represent C with an implicit function $\varphi : \Omega \to \mathbb{R}$ such that

$$\varphi(x,y) \begin{cases} > 0 & \text{if } (x,y) \in out(C) \\ = 0 & \text{if } (x,y) \in C \\ < 0 & \text{if } (x,y) \in in(C) \end{cases}. \tag{2}$$

Since this representation of C is not unique, one typically chooses $\varphi(x,y)$ to be the signed distance from (x,y) to C, i.e. $|\nabla\varphi| = 1$ almost everywhere.

The goal of Mumford-Shah segmentation is to divide Ω into regions of coherent intensities by minimizing the energy functional (1). The work of Chan and Vese [4] proposes a level set implementation of (1) in which a piecewise constant approximation $c_1 H(\varphi(x,y)) + c_2(1 - H(\varphi(x,y)))$ of $u(x,y)$ is found by minimizing

$$\begin{aligned} E(\varphi, c_1, c_2) = \mu \int_\Omega |\nabla H(\varphi(x,y))| + \lambda_1 \int_\Omega (u(x,y) - c_1)^2 (1 - H(\varphi(x,y))) \\ + \lambda_2 \int_\Omega (u(x,y) - c_2)^2 H(\varphi(x,y)), \end{aligned} \tag{3}$$

where $H(\varphi)$ is the heaviside function which is 1 if $\varphi \geq 0$ and 0 if $\varphi < 0$.

The minimization of E with respect to φ, c_1 and c_2 is usually done using an alternating minimization procedure. Assuming that c_1 and c_2 are known, one computes φ as the stationary solution of the partial differential equation (PDE)

$$\frac{\partial \varphi}{\partial t} = \delta(\varphi) \left(\mu \nabla \cdot \left(\frac{\nabla \varphi}{|\nabla \varphi|} \right) + \lambda_1 (u(x,y) - c_1)^2 - \lambda_2 (u(x,y) - c_2)^2 \right), \tag{4}$$

where $\delta(\varphi) = \frac{dH(\varphi)}{d\varphi}$. Assuming now that φ is known, the variables c_1 and c_2 are simply given by the mean intensities inside and outside of C, respectively. Iterating the updates for φ, c_1 and c_2 until convergence yields the final implicit function φ whose zero level set is the desired contour segmenting the image. This method is guaranteed to converge to a local minimum, because the cost functional is always positive and also non-increasing if the implementation of the algorithm is done carefully.

3 Dynamic Texture Segmentation

In this section, we propose a variational approach for segmenting multiple dynamic textures in an image sequence. Our algorithm is conceptually very similar to the method described in the previous section. The main difference is that instead of incorporating only image intensities in the cost functional, we also consider dynamics and texture information. Therefore, rather than finding regions of coherent intensity, we find regions of coherent dynamics and texture.

For the sake of simplicity, in Section 3.1 we assume that the boundary of the dynamic texture is static and propose a segmentation approach based solely on dynamics and intensities. We model the temporal evolution of the image intensities as the output of a mixture of ARX models whose parameters describe both the mean intensity and dynamics of each region. Under this model, we propose a generalization of Mumford-Shah segmentation to the temporal domain. Although the model does not incorporate spatial texture, the experiments will show that it is already appropriate for segmenting certain classes of dynamic textures. In Section 3.2 we extend this model to incorporate both texture and dynamics. The spatial texture at each frame is modeled with a set of second order Ising descriptors whose temporal evolution is governed by an ARX model. The segmentation algorithm minimizes an energy functional by alternating between the identification of the ARX models, which can be done linearly, and the computation of the contour using a level set implementation. In Section 3.3 we extend our segmentation approach to dynamic textures with a moving contour.

3.1 Segmentation Using Dynamics and Mean Intensity

Modeling the Dynamics: Suppose we have F frames of an image sequence $u(x, y, f)$, where $u(x, y, f)$ denotes the intensity of pixel (x, y) in the fth frame. We assume that the image intensities are the output of a mixture of ARX models of order p. That is, for each pixel (x, y), there is an ARX model j such that

$$u(x, y, f) = a_0^j + \sum_{i=1}^{p} a_i^j u(x, y, f - i) + w(x, y, f), \tag{5}$$

where $\mathbf{a}^j = [a_0^j \cdots a_p^j] \in \mathbb{R}^{1\times(p+1)}$ are the ARX parameters for region $\Omega_j \subset \mathbb{R}^2$ and $w(x, y, f) \overset{i.i.d}{\sim} N(0, \sigma^2)$ is the associated noise. Notice that this ARX model

incorporates the spatial-temporal mean intensities in the parameter a_0. The standard ARX model does not include this term, and requires subtraction of the temporal mean intensity at each pixel in order for it to be applicable. Therefore, using a standard ARX model would only be useful in detecting differences in dynamics, but would not be able to deal with dynamic textures with different mean intensities. Recall that the method used in [8] uses standard ARMA models, which also require the subtraction of the temporal mean intensities.

Variational Method of Segmentation: In order to segment the video sequence according to the different ARX models, we propose a spatial-temporal extension of the level set approach described in Section 2. We replace the last two terms in (3) by the spatial-temporal mean squared prediction error to model (5):

$$\begin{aligned} E =& \mu \int_{\Omega} |\nabla H(\varphi(x,y))| dxdy \\ &+ \lambda_1 \int_{\Omega} \sum_{f=p+1}^{F} [(u(x,y,f) - c_1(x,y,f))^2](1 - H(\varphi(x,y)))dxdy \\ &+ \lambda_2 \int_{\Omega} \sum_{f=p+1}^{F} [(u(x,y,f) - c_2(x,y,f))^2] H(\varphi(x,y))dxdy, \end{aligned} \tag{6}$$

where

$$c_j(x,y,f) = a_0^j + \sum_{i=1}^{p} a_i^j u(x,y,f-i) \qquad j = 1,2. \tag{7}$$

This definition for the cost functional makes intuitive sense, because if w is zero mean Gaussian noise and $\lambda_1 = \lambda_2 = \lambda$, then E can be written as

$$E = \mu \int_{\Omega} |\nabla H(\varphi(x,y))| dxdy + \lambda \int_{\Omega} \sum_{f=p+1}^{F} w(x,y,f)^2 dxdy. \tag{8}$$

Therefore, the last term is simply the spatial-temporal variance of the noise $w(x,y,f)$. This observation makes our cost function both algebraically and statistically meaningful, unlike the cost functional in [8], which depends only on the algebraic properties of the ARMA models.

In order to minimize E in (6), we generalize the classical Mumford-Shah segmentation method described in Section 2. Assuming that $\mathbf{a}^1$ and $\mathbf{a}^2$ are known, we solve for the implicit function φ as the stationary solution of the PDE:

$$\begin{aligned} \frac{\partial \varphi}{\partial t} = \delta(\varphi)\Big(\mu \nabla \cdot \big(\frac{\nabla \varphi}{|\nabla \varphi|}\big) &+ \lambda_1 \sum_{f=p+1}^{F} (u(x,y,f) - c_1(x,y,f))^2 \\ &- \lambda_2 \sum_{f=p+1}^{F} (u(x,y,f) - c_2(x,y,f))^2\Big). \end{aligned} \tag{9}$$

Given φ, we need to update the ARX parameters $\mathbf{a}^1$ and $\mathbf{a}^2$ from the intensities of the pixels within Ω_1 and Ω_2, where Ω_1 and Ω_2 are the regions inside and outside the contour C, respectively. The problem of identifying the parameters of a single-input single-output (SISO) ARX model is a standard system identification problem [21]. Since in our problem each region has multiple pixels, we simply need to extend the standard identification methods to multiple outputs, which we do by minimizing E with respect to $\mathbf{a}^1$ and $\mathbf{a}^2$. Setting the partial derivatives of E with respect to $\mathbf{a}^1$ and $\mathbf{a}^2$ to zero leads to the following system of linear equations:

$$\mathbf{a}^j U_f^j = \mathbf{b}_f^j \qquad j = 1, 2 \quad \text{and} \quad f = p+1, \cdots, F. \tag{10}$$

The matrix U_f and the vector $\mathbf{b}_f$ are given by

$$U_f^j = \begin{bmatrix} 1 & \cdots & 1 \\ u(x_1^j, y_1^j, f-1) & \cdots & u(x_{k_j}^j, y_{k_j}^j, f-1) \\ \vdots & & \vdots \\ u(x_1^j, y_1^j, f-p) & \cdots & u(x_{k_j}^j, y_{k_j}^j, f-p) \end{bmatrix}_{(p+1)\times k_j} \qquad j = 1, 2 \tag{11}$$

$$\mathbf{b}_f^j = \begin{bmatrix} u(x_1^j, y_1^j, f) & \cdots & u(x_{k_j}^j, y_{k_j}^j, f) \end{bmatrix}_{1\times k_j} \qquad j = 1, 2, \tag{12}$$

where $\{(x_1^j, y_1^j), \cdots, (x_{k_j}^j, y_{k_j}^j)\}$ is the set of pixels in Ω_j for $j = 1, 2$. The least squares solution to the above system of linear equations is given by:

$$\mathbf{a}^j = \Big(\sum_{f=p+1}^{F} \mathbf{b}_f^j (U_f^j)^\top\Big)\Big(\sum_{f=p+1}^{F} U_f^j (U_f^j)^\top\Big)^{-1} \qquad j = 1, 2. \tag{13}$$

Iterating the updates (9) and (13) until convergence of the implicit function φ and the ARX parameters $\mathbf{a}^1$ and $\mathbf{a}^2$, leads to the final contour, which is given by the zero level set of φ.

3.2 Segmentation Using Both Dynamics and Texture

In this section, we incorporate static texture information into the segmentation process. The first step is to extract the texture information into a feature vector. We then treat this feature vector in the same fashion as we treated the pixel intensities in the previous method. In this way, we are able to incorporate the mean texture information of the regions instead of the mean pixel intensities.

Representing Static Textures: We choose the Ising second order model [22] to represent the static texture at pixel (x, y) and frame f with a five dimensional feature vector $\mathbf{u}(x, y, f) \in \mathbb{R}^5$. In this method, one chooses a neighborhood W of size $w \times w$ around each pixel, and considers the set of all cliques of type $i = 1, \ldots, 4$, C_i, where the four types of cliques are shown in Figure 1.

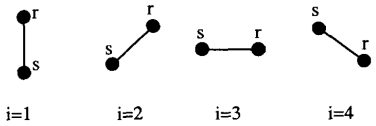


Fig. 1. The 4 clique types of the Ising second order model.

For each clique $c = (\mathbf{r}, \mathbf{s}) \in C_i$, $i = 1, \ldots, 4$, one defines the function

$$\Delta_c(x, y, f) = \begin{cases} -1 & \text{if } |u(\mathbf{r}, f) - u(\mathbf{s}, f)| < \epsilon \\ +1 & \text{otherwise} \end{cases}, \tag{14}$$

where $\epsilon > 0$ is a user specified parameter. The ith entry of the texture descriptor is defined as

$$\mathbf{u}_i(x, y, f) = \begin{cases} \sum\limits_{c \in C_i} \Delta_c(x, y, f) & \text{if } i \neq 5 \\ \frac{1}{w^2} \sum\limits_{(x,y) \in W} u(x, y, f) & \text{if } i = 5 \end{cases}. \tag{15}$$

Note that the last entry of $\mathbf{u}$ is simply the mean intensity in a neighborhood of size w around each pixel.

Modeling the Temporal Evolution of the Texture Descriptors: To model different dynamic textures, we take a similar approach to the one described in Section 3.1. But instead of working with the intensities alone, we work with the feature vectors. We assume that the texture descriptors are the output of a mixture of ARX models of order p. That is, for each pixel (x, y), there is an ARX model j such that

$$\mathbf{u}(x, y, f) = \mathbf{a}_0^j + \sum_{i=1}^{p} A_i^j \mathbf{u}(x, y, f - i) + \mathbf{w}(x, y, f), \tag{16}$$

where $\mathbf{a}_0^j \in \mathbb{R}^5$ is now the mean texture vector and $A_1^j, \cdots, A_p^j \in \mathbb{R}^{5 \times 5}$ are the ARX parameter matrices of region Ω_j. Notice that including the parameter $\mathbf{a}_0$ allows us to incorporate the mean textures of a region in our algorithm, the same way including a_0 in (5) allowed us to incorporate mean intensities.

Variational Method of Segmentation: The segmentation approach is a modified version of the one described in Section 3.1. The difference is that we now work with the texture descriptors instead of the intensities. Therefore, the

cost functional is modified as

$$\begin{aligned} E =& \mu \int_{\Omega} |\nabla H(\varphi(x,y))| dxdy \\ &+ \lambda_1 \int_{\Omega} \sum_{f=p+1}^{F} \|\mathbf{u}(x,y,f) - \mathbf{c}_1(x,y,f)\|^2 (1 - H(\varphi(x,y))) dxdy \\ &+ \lambda_2 \int_{\Omega} \sum_{f=p+1}^{F} \|\mathbf{u}(x,y,f) - \mathbf{c}_2(x,y,f)\|^2 H(\varphi(x,y)) dxdy, \end{aligned} \tag{17}$$

where

$$\mathbf{c}_j(x,y,f) = \mathbf{a}_0^j + \sum_{i=1}^{p} A_i^j \mathbf{u}(x,y,f-i) \qquad j = 1,2 \tag{18}$$

with $\mathbf{a}_0^j$ and A_i^j the ARX parameters associated with the pixels in region Ω_j.

Given the ARX model parameters, the update formula for the embedding function φ is given by

$$\begin{aligned} \frac{\partial \varphi}{\partial t} = \delta(\varphi)\Big(\mu \nabla \cdot \big(\frac{\nabla \varphi}{|\nabla \varphi|}\big) &+ \lambda_1 \sum_{f=p+1}^{F} \|\mathbf{u}(x,y,f) - \mathbf{c}_1(x,y,f)\|^2 \\ &- \lambda_2 \sum_{f=p+1}^{F} \|\mathbf{u}(x,y,f) - \mathbf{c}_2(x,y,f)\|^2\Big). \end{aligned} \tag{19}$$

Given φ, the update formula for the ARX parameters is given by

$$\big[\mathbf{a}_0^j \; A_1^j \cdots A_p^j\big] = \Big(\sum_{f=p+1}^{F} B_f^j (U_f^j)^\top\Big)\Big(\sum_{f=p+1}^{F} U_f^j (U_f^j)^\top\Big)^{-1} \qquad j = 1,2, \tag{20}$$

which is an extended version of equation (13). The matrices U_f^j and B_f^j are built from the pixel feature vectors as follows

$$U_f^j = \begin{bmatrix} 1 & \cdots & 1 \\ \mathbf{u}(x_1^j, y_1^j, f-1) & \cdots & \mathbf{u}(x_{k_j}^j, y_{k_j}^j, f-1) \\ \vdots & & \vdots \\ \mathbf{u}(x_1^j, y_1^j, f-p) & \cdots & \mathbf{u}(x_{k_j}^j, y_{k_j}^j, f-p) \end{bmatrix}_{(5p+1)\times k_j} \tag{21}$$

$$B_f^j = \Big[\mathbf{u}(x_1^j, y_1^j, f) \cdots \mathbf{u}(x_{k_j}^j, y_{k_j}^j, f)\Big]_{(5p+1)\times k_j}, \tag{22}$$

where the pixels $\{(x_1^j, y_1^j), \cdots, (x_{k_j}^j, y_{k_j}^j)\}$ belong to region Ω_j. We then update φ and the ARX parameters in an iterative manner until convergence.

In our implementation, we assume that the ARX parameter matrices are diagonal. This implies that the five entries of the feature vector $\mathbf{u}(x,y,f)$ are

the outputs of five decoupled scalar ARX models. Therefore, we can estimate the ARX parameters independently for each entry of $\mathbf{u}$. This assumption significantly reduces the computational complexity of the method.

3.3 Segmentation of Dynamic Textures with Moving Boundaries

So far, we have only talked about dynamic textures with fixed boundaries. We now consider the case in which the boundaries of the regions also vary with time. In order to track the moving boundaries, we apply the method described in Section 3.2 to a moving window of frames in time, which is of size $F > p$. More specifically, the algorithm works as follows:

1. Given a user specified embedding φ_0 representing an initial contour, apply the segmentation method described in Section 3.2 to frames $1, \ldots, F$. This results in a new embedding, φ_1.
2. Use φ_1 as an initial embedding, and apply the segmentation method of Section 3.2 to frames $2, \ldots, F+1$. This results in a new embedding φ_2.
3. Repeat the previous step for all the remaining frames of the sequence.

Therefore, at every frame f, we have an embedding φ_f, which in turn gives us a contour C_f. In this way, we are able to follow the moving boundaries of the regions, provided that the sampling frequency of the video is high enough, so that the boundaries do not move significantly in the F adjacent frames.

4 Experimental Results

In this section, we present experiments performed on various types of sequences using the different methods introduced in the previous sections. We first compare the methods described in this paper to each other, to emphasize the essence of every step involved in the development of our dynamic texture segmentation method. We then compare the results of our most general method to those from the original implementation of the existing methods used in [8] and [16]. We take the sequences from [8] and [16] in order to make fair comparisons among the methods. Finally, we present results on a real sequence taken from a raccoon caught on a river. For all sequences, the pixel intensities are normalized between 0 and 1, and the orders of the ARX models are manually set to $p = 2$.

4.1 Comparison of the Methods Introduced in this Paper

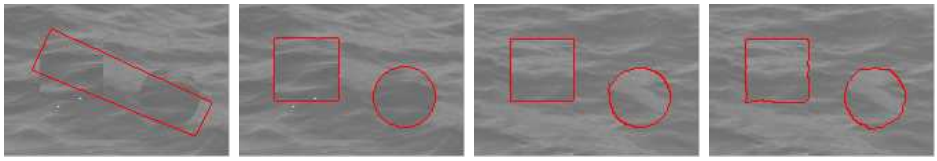
In this section, we compare the performance of the following methods:

1. *Method 1*: The method introduced in Section 3.1 with a_0 set to zero. This method only detects differences in the dynamics of the regions, and requires the temporal mean intensity at each pixel to be subtracted from the sequence.
2. *Method 2*: The method introduced in Section 3.1. This method detects differences in the dynamics or mean spatial-temporal intensities of the regions.

3. *Method 3*: The method introduced in Section 3.2 with parameters $w = 3$ pixels and $\epsilon = 0.02$. This method successfully segments dynamic textures.

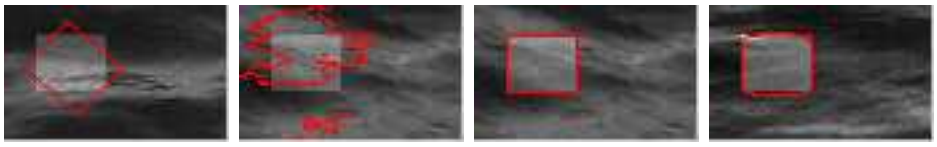
Figure 2 shows the performance of the methods on the `ocean_dynamics` sequence. This is a video sequence of the ocean in which the regions in the circle and the square move twice as fast as the background. Thus, the two dynamic textures are identical in appearance, but differ in dynamics. Notice that all the methods successfully segment the sequence. This is expected, since all the methods incorporate dynamics.

Figure 3 shows the performance of the methods on the `ocean_intensity` sequence, in which the region in the square has a higher mean intensity than the background. Thus, the dynamic textures are identical in dynamics, but differ in mean intensity. We see that the first method fails to correctly segment the regions. This is expected, since Method 1 can only detect variations in dynamics. On the other hand, methods 2 and 3 are successful, since they incorporate appearance as well as dynamics. Notice that Method 3 gives smoother results than Method 2, because the last element of the feature vector $\mathbf{u}$ uses a smoothened version of the image intensities in a spatial neighborhood of size 3×3.



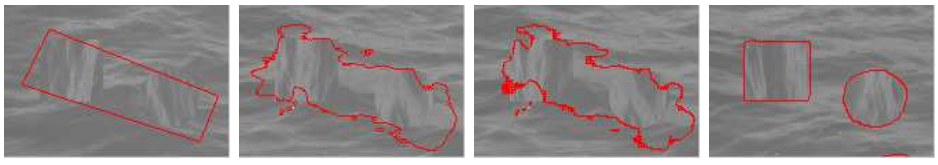
(a) Initial contour (b) Method 1 (c) Method 2 (d) Method 3

Fig. 2. Results of Methods 1-3 on the `ocean_dynamics` sequence.



(a) Initial contour (b) Method 1 (c) Method 2 (d) Method 3

Fig. 3. Results of Methods 1-3 on the `ocean_intensity` sequence.



(a) Initial contour (b) Method 1 (c) Method 2 (d) Method 3

Fig. 4. Results of Methods 1-3 on the `ocean_appearance` sequence.

Figure 4 shows the performance of the methods on the `ocean_appearance` sequence. This is a video sequence of the ocean in which the regions in the square and the circle have been rotated by 90 degrees. Thus, the dynamic textures differ only in the texture orientation, but share the same dynamics and general appearance (grayscale values). We see that only the last method is able to correctly segment this sequence. The other methods fail because both the dynamics and mean intensities of the two regions are the same.

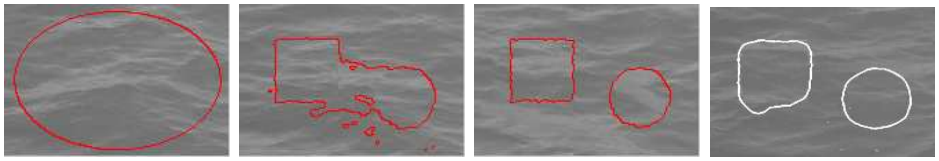
4.2 Fixed Boundary Dynamic Texture Segmentation Results and Comparison

In this section, we show the results of our most general method (Method 3) on various sequences containing dynamic textures with fixed boundaries, and compare them to those of state-of-the-art methods. Figures 5 to 7 show a comparison of our method and the method used in [8], using the same sequences and initializations as in [8]. Figure 5 shows results on the `ocean_dynamics` sequence. Figure 6 shows the results on the `ocean_appearance` sequence. Figure 7 shows the results on the `ocean_smoke` sequence which contains both smoke and sea water. Notice that our method gives a more accurate segmentation than the method in [8], even though we use a significantly smaller number of frames and a simpler dynamical model of significantly lower order for each region. In [8], the order of the ARMA model for each region is $p = 10$ and the number of frames is $F = 120$ on all the sequences, whereas we use ARX models of order $p = 2$, and $F = 20$ in our method.

4.3 Moving Boundary Dynamic Texture Segmentation Results and Comparison

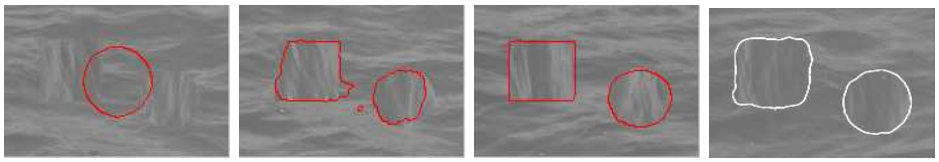
An important improvement of our method over the method in [8] is that we can handle regions with moving boundaries. This is mainly because our method can successfully segment dynamic textures using a significantly smaller number of frames and a much lower order for the ARX models. We present the performance of our method for moving boundaries using the same sequence and initialization as in [8]. We choose the order of the regions to be $p = 2$ and the temporal window size to be $F = 5$ for this sequence. Figure 8 shows the results on the `ocean_fire` sequence containing both fire and sea water. One can see that our method can track the moving boundary of the fire, whereas in [8] the boundary is not successfully detected even in one frame.

We also compare our results against those from [16]. Figure 9 shows the `ocean_grass` sequence in which the region inside the square is taken from grass moving with the wind, and the background is sea water. The order of the regions is again $p = 2$, and the temporal window size is $F = 5$ in our method. Notice that our algorithm is generally successful, but makes some mistakes at the top left corner of the sequence. The first reason for this is that the water does not have a uniform texture or dynamics all over the image plane; in fact, the top left corner is darker and moves more slowly compared to the other parts of the water.



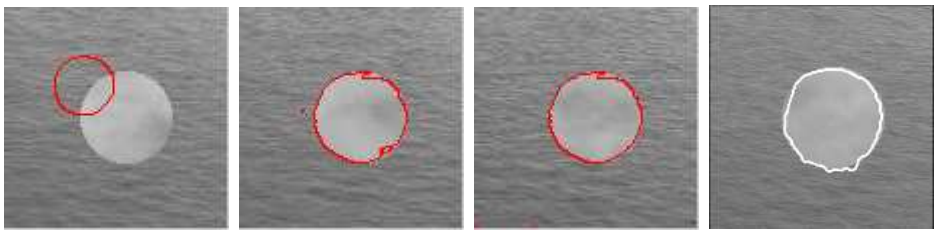
(a) Initial contour (b) Intermediate step (c) Final contour (d) Result from [8]

Fig. 5. (a)-(c) The result of our dynamic texture segmentation method on the `ocean_dynamics` sequence. (d) The result from [8] for the same sequence.



(a) Initial contour (b) Intermediate step (c) Final contour (d) Result from [8]

Fig. 6. (a)-(c) The result of our dynamic texture segmentation method on the `ocean_appearance` sequence. (d) The result from [8] for the same sequence.



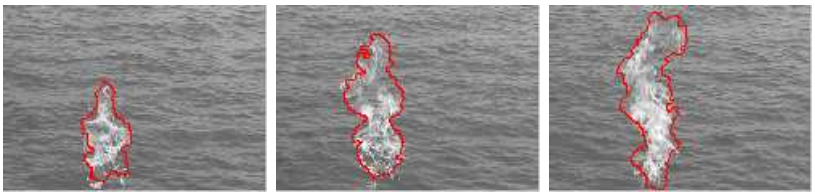
(a) Initial contour (b) Intermediate step (c) Final contour (d) Result from [8]

Fig. 7. (a)-(c) The result of our dynamic texture segmentation method on the `ocean_smoke` sequence. (d) The result from [8] for the same sequence.

The second reason is that the grass region does not have significant dynamics to it. Therefore, that specific part of the water region is similar to the grass in both appearance and dynamics. Notice also that the method in [16] gives a bigger rectangle than the true boundary of the grass region, whereas our result is closer to the true boundary. However, the method in [16] does not make the mistake at the upper left corner of the sequence. This is because it only incorporates dynamics, and does not take appearance into account.

4.4 Experimental Results on a Real Sequence

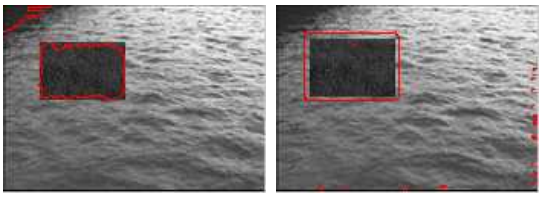
In this section, we present the results of our moving boundary dynamic texture segmentation algorithm applied to a video taken from a raccoon caught on a



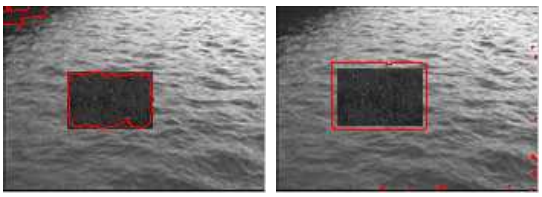
(a) Method 3, frame 5 (b) Method 3, frame 15 (c) Method 3, frame 30

(d) Method 3, frames 40 (e) Method 3, frame 70 (f) Method in [8]

Fig. 8. (a)-(e) Results of our moving boundary segmentation method on various frames of the `ocean_fire` sequence. (f) The result from [8] for the same sequence.



(a) Method 3 (b) Method in [16]



(c) Method 3 (d) Method in [16]

Fig. 9. (a),(c) Results of our moving boundary segmentation method on two frames of the `ocean_grass` sequence. (b),(d) Results from [16] for the same sequence.

river. The sequence contains 100 frames in total. The size of the temporal window is chosen as $F = 5$ and the order of the ARX models is chosen to be $p = 2$.

Figure 10 shows the performance of the algorithm on tracking the boundary of the raccoon throughout the sequence. Notice that the algorithm performs fairly well on this challenging sequence.

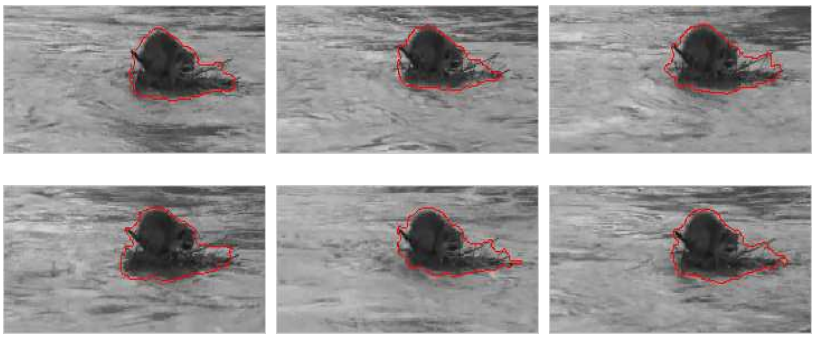


Fig. 10. Results of our moving boundary segmentation method on various frames of the raccoon sequence.

5 Conclusions and Future Work

We have introduced a new method for segmenting dynamic textures that combines Ising texture descriptors, ARX dynamical models and level set methods. In spite of its simplicity, our experiments showed that our method performs better than existing algebraic and variational approaches to dynamic texture segmentation, not only in terms of speed and accuracy, but also in its ability to track regions with moving boundaries in a sequence.

However, we have only used manually set orders for the dynamical models associated with each region. Moreover, different regions in a sequence are modeled with the same order. Future work includes combining our method with model selection techniques for automatic order selection.

Acknowledgements

This work was partially supported by Hopkins WSE startup funds, and by grants NSF CAREER IIS-0447739, NSF CRS-EHS-0509101, NIH-NHLBI 5-R01-HL-64795-01, and ONR N00014-05-1-0836.

References

1. Mumford, D., Shah, J.: Boundary detection by minimizing functionals. IEEE Conference on Computer Vision and Pattern Recognition (1985) 22–26
2. Witkin, A., Terzopulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1** (1987) 321 – 331
3. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. In: IEEE Conference on Computer Vision and Pattern Recognition. (1995) 694 – 699
4. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Transactions on Image Processing **10** (2001) 266–277

5. Tsai, A., Jr., A.Y., Willsky, A.: Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification. IEEE Transactions on Image Processing **10** (2001) 1169 – 1186
6. Paragios, N., Deriche, R.: Geodesic active contours and level set methods for supervised texture segmentation. International Journal of Computer Vision **46** (2002) 223–247
7. Cremers, D., Soatto, S.: Motion competition: A variational framework for piecewise parametric motion segmentation. International Journal of Computer Vision **62** (2005) 249–265
8. Doretto, G., Cremers, D., Favaro, P., Soatto, S.: Dynamic texture segmentation. In: IEEE Conference on Computer Vision. (2003) 44–49
9. Doretto, G., Chiuso, A., Wu, Y., Soatto, S.: Dynamic textures. International Journal of Computer Vision **51** (2003) 91–109
10. Yuan, L., Wen, F., Liu, C., Shum, H.Y.: Synthesizing dynamic texture with closed-loop linear dynamic system. In: European Conference on Computer Vision. (2004) 603–616
11. Moor, B.D., Overschee, P.V., Suykens, J.: Subspace algorithms for system identification and stochastic realization. Technical Report ESAT-SISTA Report 1990-28, Katholieke Universiteit Leuven (1990)
12. Soatto, S., Doretto, G., Wu, Y.: Dynamic textures. In: IEEE International Conference on Computer Vision. (2001) 439–446
13. Doretto, G., Soatto, S.: Editable dynamic textures. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume II. (2003) 137–142
14. Saisan, P., Doretto, G., Wu, Y.N., Soatto, S.: Dynamic texture recognition. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume II. (2001) 58–63
15. Fitzgibbon, A.: Stochastic rigidity: Image registration for nowhere-static scenes. In: IEEE International Conference on Computer Vision. (2001) 662–669
16. Vidal, R., Ravichandran, A.: Optical flow estimation and segmentation of multiple moving dynamic textures. In: IEEE Conference on Computer Vision and Pattern Recognition. Volume II. (2005) 516–521
17. Chan, A., Vasconcelos, N.: Mixtures of dynamic textures. In: IEEE International Conference on Computer Vision. Volume 1. (2005) 641 – 647
18. Cooper, L., Liu, J., Huang, K.: Spatial segmentation of temporal texture using mixture linear models. In: Proceedings of the Dynamical Vision Workshop in the International Conference of Computer Vision. (2005)
19. Vidal, R., Ma, Y., Sastry, S.: Generalized Principal Component Analysis (GPCA). IEEE Trans. on Pattern Analysis and Machine Intelligence **27** (2005) 1–15
20. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: Algorithms based on hamilton–jacobi formulations. **79** (1988) 12–49
21. Ljung, L.: System Identification: theory for the user. Prentice Hall (1987)
22. Roula, M.A., Bouridane, A., Amira, A., Sage, P., Milligan, P.: A novel technique for unsupervised texture segmentation. In: IEEE International Conference on Image Processing. Volume 1. (2001) 58 – 61