

TEXTURA DINÂMICA

Dalia Melissa Bonilla Correa

27 de Novembro de 2006

Resumo

Textura dinâmica também conhecida em inglês como **temporal texture** ou **video texture** [2]. É uma seqüência de imagens de cenas de movimento que tem propriedades estacionárias no tempo [4]. Esta seqüência de imagens varia continuamente e infinitamente, e uma cena individual pode ser repetida de tempo em tempo [3]. Exemplos são as ondas do oceano, fumaça, fontes de água, fogo, escadas rolantes, chamas de vela, etc. As texturas dinâmicas usadas para personalizar páginas webs, jogos de computadores e protetores de tela [2].

Presentarei a continuação um resumo, análise e implementação sobre o artigo *Synthesizing Dynamic Texture with Closed-Loop Linear Dynamic System*[2], e uma breve explicação de como se entende matematicamente uma textura dinâmica.

1 Processo:

Gerar uma nova seqüência que é similar mas algo diferente de uma seqüência de vídeo original. O vídeo gerado pode ser rodado infinitamente sem descontinuidades visíveis.

Observação: uma textura dinâmica é fácil de armazenar e calcular.

Consideremos uma seqüência de imagens de uma cena de movimento. Cada imagem é uma matriz de números positivos que será analisada como um sinal visual. Logo se interpreta e entende uma quantidade de sinais para inferir um modelo estocástico que o gera. Além o modelo deve prever futuros sinais com grau de precisão máxima (probabilidade mínimo erro de previsão)[4].

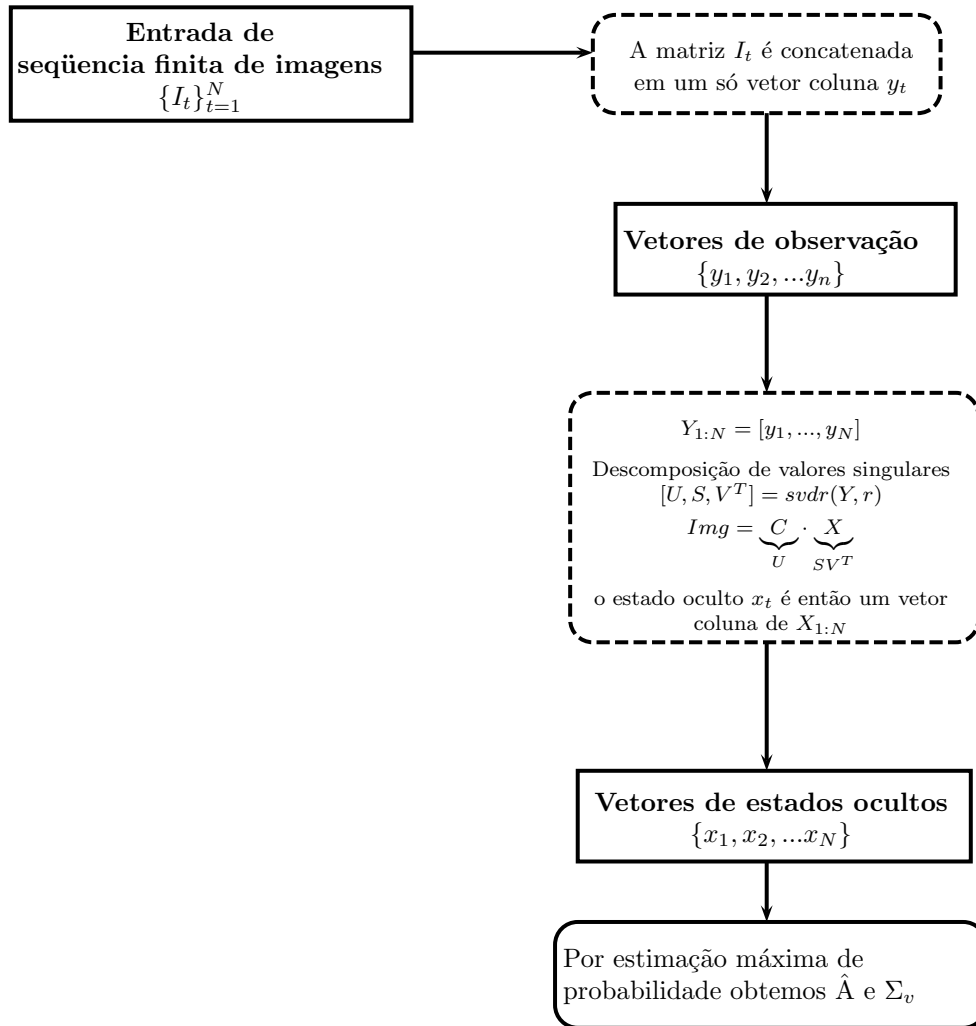
O processo tem uma parte de análise ou treinamento, onde o modelo identifica parâmetros de dados observados e uma parte de síntese onde gera resultados do aprendizado.

Soatto propor [4] um sistema dinâmico linear (LDS) para analisar uma textura dinâmica. O modelo é

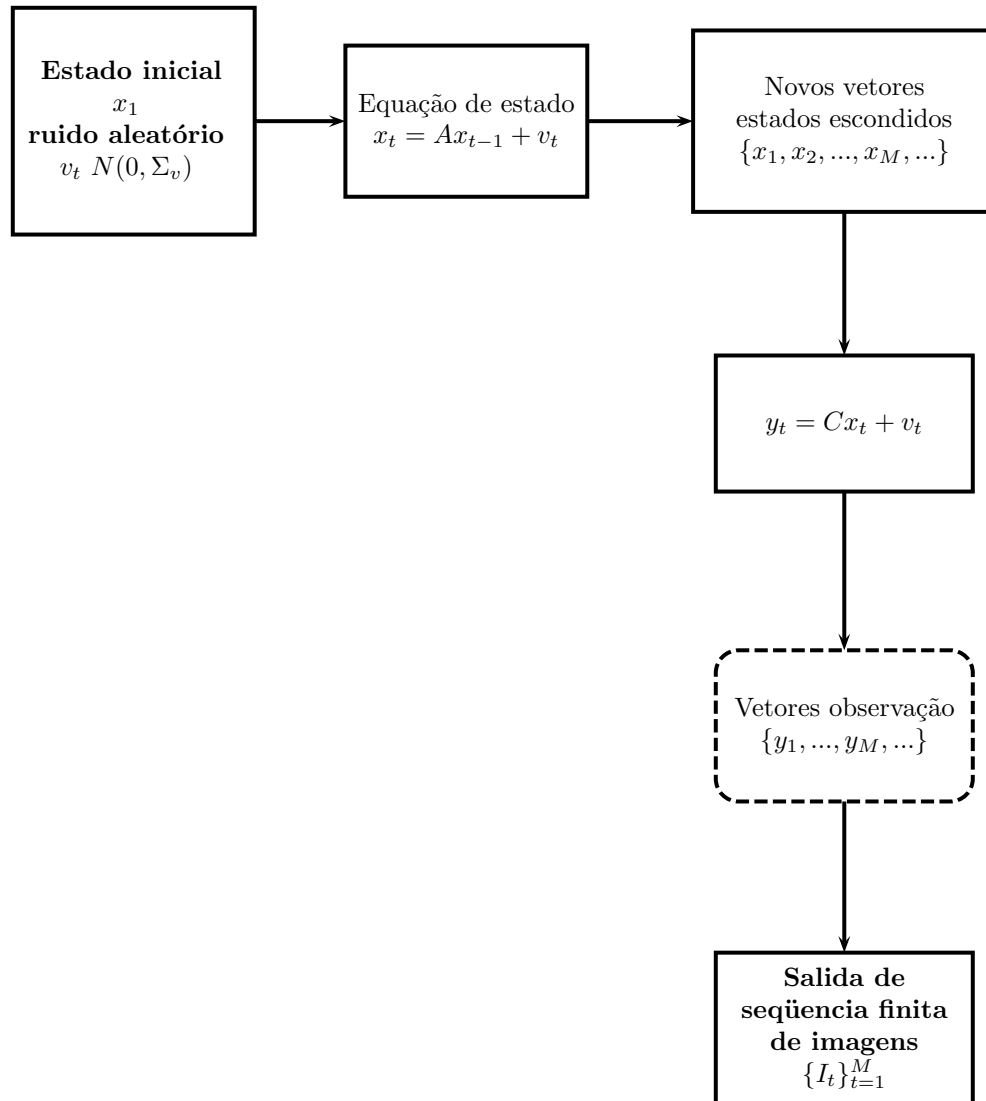
$$\begin{cases} x_t = Ax_{t-1} + v_t, & v_t \sim N(0, \Sigma_v) \\ y_t = Cx_t + w_t, & w_t \sim N(0, \Sigma_w) \end{cases}$$

Onde $y_t \in R^n$ é um vetor de observação e $x_t \in R^r$, $r \ll n$ é um estado desconhecido, A matriz de sistema, C matriz de saída e v_t, w_t ruídos gaussianos.

2 Treinamento de LDS



3 Sínteses



Então substituindo u_t em \triangleright temos a seguinte equação

$$x_t = \sum_{i=-p}^{-1} \phi_{p+1+i} x_{t+i} + \sum_{j=1}^q \phi_{p+j} x_{t+j} + v_t \quad v_t \sim N(0, \Sigma_v)$$

onde ϕ_k é uma combinação linear de A_i e D_j .

Fazendo

$$\theta = [\phi_1, \dots, \phi_p, \phi_{p+1}, \dots, \phi_{p+q}]^\top$$

$$\varphi(t) = [x_{t-p}^\top, \dots, x_{t-1}^\top, x_{t+1}^\top, \dots, x_{t+q}^\top]^\top$$

Então podemos escrever a equação desta forma sem o ruído $x_t = \theta^\top \varphi(t)$ e agora nosso problema é minimizar

$$\|x_t - \theta^\top \varphi(t)\|_2^2$$

e por estimativa de mínimos quadrados (LSE)[1] temos a solução do problema é a seguinte matriz

$$\hat{\theta}_N = \left[\sum_{t=1}^N \varphi(t) \varphi^\top(t) \right]^{-1} \sum_{t=1}^N \varphi(t) x^\top(t)$$

Seja $R = \sum_{t=1}^N \varphi(t) \varphi^\top(t)$ e temos então que

$$[R]_{ij} = \sum_{t=1}^N x_{t+i} x_{t+j}^\top, \quad -p \leq i, j \leq q$$

Nosso caso só conhecemos x_t para $1 \leq t \leq N$, mas podemos fazer uma extensão para os valores de $t = -p, \dots, -1, 0$ e para $t = N+1, \dots, N+q$ considerando $x_t = 0$. Logo pelo método de autocorrelação temos

$$[R]_{ij} = \sum_{t=1-i}^{N-j} x_{t+i} x_{t+j}^\top, \quad -p \leq i, j \leq q$$

Agora adotaremos a seguinte notação

$$G_i = [R_{i,-p}, \dots, R_{i,-1}, R_{i,1}, \dots, R_{i,q}]$$

e é fácil verificar que

$$G_0^\top = \left[\sum_{t=1}^N \varphi(t) x^\top(t) \right] \text{ e } R = [G_{-p}^\top, \dots, G_{-1}^\top, G_1^\top, \dots, G_q^\top]^\top.$$

Portanto temos que

$$\theta = \hat{\theta}_N = R^{-1}G_0^\top \text{ ou}$$

$$\theta^\top = G_0 [R^\top]^{-1}$$

$$\boxed{[\phi_1, \dots, \phi_p, \phi_{p+1}, \dots, \phi_{p+q}] = G_0 [G_{-p}^\top, \dots, G_{-1}^\top, G_1^\top, \dots, G_q^\top]^{-1}}$$

Onde

$$\Sigma_v = \frac{1}{N - p - q} \left(R_{0,0} - \sum_{i=-p}^q \phi_{i+p+1} R_{i,0} \right)$$

e a matriz covariância.

Em resumo temos o seguinte

$$[\phi_1, \dots, \phi_p, \phi_{p+1}, \dots, \phi_{p+q}] = G_0 [G_{-p}^\top, \dots, G_{-1}^\top, G_1^\top, \dots, G_q^\top]^{-1}$$

$$\Sigma_v = \frac{1}{N - p - q} \left(R_{0,0} - \sum_{i=-p}^q \phi_{i+p+1} R_{i,0} \right)$$

$$G_i = [R_{i,-p}, \dots, R_{i,-1}, R_{i,1}, \dots, R_{i,q}],$$

$$R_{i,j} = \sum_{t=1-i}^{N-j} x_{t+i} x_{t+j}^\top, \text{ para } -p \leq i, j \leq q$$

4 Sínteses usando CLDS:

Usando os vetores estados ocultos $\{x\}_{t=1}^N$ gerados da seqüência original de imagens $\{x\}_{t=1}^N$, calculamos as similaridades de estado a estado e são guardadas na matriz $S = \{s_{i,j}\}$ onde $s_{i,j} = \frac{\langle x_i, x_j \rangle}{\sqrt{\langle x_i, x_i \rangle \langle x_j, x_j \rangle}}$. Definamos a matriz de transcrição P através de uma função exponencial

$$P_{i,j} = P = \begin{cases} \exp\{\gamma s_{i+1,j}\}, & i \neq j \\ 0, & i = j \end{cases}$$

P é normalizada tal que $\sum_j P_{ij} = 1$ e γ é um numero de 1 a 50. Devemos escolher agora uma seqüência inicial. Geramos uma seqüência

$$\{x_{k_1:k_1+\tau-1}, x_{k_2:k_2+\tau-1}, \dots, x_{k_h:k_h+\tau-1}\}$$

de h clips onde $x_{k_i:k_i+\tau-1}$ é um clipe curto de τ frames para $i = 1, \dots, h$ e os clips são tais que $x_{k_i+\tau-1}$ e $x_{k_{i+1}}$ estão muito perto segundo P .

4.1 Algoritmo

1. Para sintetizar M frames, fixamos o tamanho do clipe τ e escolhemos aleatoriamente de $\{x\}_{t=1}^N$ o primeiro dos clips $\{x_{k_1}, x_{k_1+1}, \dots, x_{k_1+\tau-1}\}$.
2. Amostramos o seguinte clipe $\{x_{k_2}, x_{k_2+1}, \dots, x_{k_2+\tau-1}\}$ com $P(x_{k_2}|x_{k_1+\tau-1})$ em P .
3. Repetir 2, ate os clips h são amostrados.
4. ($M \leq h \times \tau$) Os M primeiros estados são tomados como sequência inicial $\{x_1^{(0)}, x_2^{(0)}, \dots, x_M^{(0)}\}$
5. Iteramos $n = 1, 2, \dots$ ate $|\delta^{(n)} - \delta^{(n-1)}| < \varepsilon$

(a) Amostramos o ruído $v_t^{(n)}$ e $x_t^{(n)}$ como segue

$$x_t = \sum_{i=-p}^{-1} \phi_{p+1+i} x_{t+i} + \sum_{j=1}^q \phi_{p+j} x_{t+j} + v_t \quad v_t \sim N(0, \Sigma_v)$$

para $t = 1, 2, \dots, M$.

(b) Calculamos $\delta^{(n)}$

$$\delta^{(n)} = \sum_{t=1}^M \left\| x_t^{(n)} - x_t^{(n-1)} \right\|_2.$$

6. Saída da sequência final $\{x_1^{(n)}, x_2^{(n)}, \dots, x_M^{(n)}\}$

5 Implementação

O software usado para a implementação foi MATALAB e fiz algumas modificações de códigos fornecidos por Anderson Mayrink da Cunha, IMPA, 2006.

5.1 Algoritmo e Resultados

O algoritmo esta dividido em uma parte de treinamento:

1. Calculos dos vetores estados ocultos x_t
2. Calculo dos parametros do CLDS que nosso caso são os ϕ
3. Calculo da matriz Σ_v

E tambem uma parte de sintesis

1. Calculo da matriz P probabilidade de transicao
2. Calculo da sequencia nova

O código do algoritmo é o seguinte:
 Cálculo de estados ocultos da sequência original de imagens.

```
[m,T]= size(Y);

if mean_Y
    clds.Ymean= mean(Y,2);    % mean like Doretto - each pixel
else
    clds.Ymean= zeros(m,1);
end
% Obs. model
[U,S,V]= svds(Y- clds.Ymean*ones(1,T), n);
clds.C= U;
clds.X=S*V';
```

Síntesis de CLDS

```
n=1;% o estado inicial é 1
%e=1/10;
k=1;
while (k>e) %fazer até que k<e
    n=n+1;
    F(n)=0;
    VN=MC*randn(L,M); % para cada n uma v.a aleatoria
    for t=1:M
        %z=size(m.PHI(:,:,2)*E(:,1,1));
        S=zeros(L,1);
        for i=-p:-1

            if ((t+i)> 0)

                S = S + m3.PHI(:,:,p+1+i)*E(:,t+i,n);
                %coordenada é substituída por t já que as linhas
                %estão concatenadas de 1 até M
            end
        end
        Z=zeros(L,1);    %size(m.PHI(:,:,1+1)*E(:,1,1));
        for j=1:q
            if ((t+j)<=M) % M novos frames
                C=m3.PHI(:,:,1+j);
                size(C)
                size(E(:,t+j,n-1))
                Z = Z + C*E(:,t+j,n-1);
            else
                Z=Z;
            end
        end
        E(:,t,n)=S+Z+VN(:,t); %formula 6 pag 610
```

```

Q=E(:,t,n);
W=E(:,t,n-1);

% modificacao do erro 27/nov/2006

no=(norm(Q-W))^2; % norma ao quadrado

F(n)= F(n) + (1/(M*L))*no;      % calculo delta
end
if (n>=3)
    k=abs(F(n)-F(n-1));
else
    k=abs(F(n)); %k=e+1; % para n<2 repetir
end
end
for i=1:M
    I(:,i)= clds.C*E(:,i,n) + clds.Ymean;
end

```

Os resultados obtidos foram os seguintes:

- Para um video original de prueba de 12.8KB, de 30 frames, obtemos com um video de salida de tambem 30 frames e parametro $\tau = 5$.

```

estado.m: Learning took 0.91097 seconds.
modelo.m: Learning took 0.076879 seconds.
modelo.m: Learning took 0.028147 seconds.
matriz.m: Learning took 0.016425 seconds.
sintesis.m: Learning took 1.0643 seconds.

```

- Para o mesmo video agora criando o dobro de frames e parametro $\tau = 5$.

```

estado.m: Learning took 0.12708 seconds.
modelo.m: Learning took 0.008682 seconds.
modelo.m: Learning took 0.014162 seconds.
matriz.m: Learning took 0.001276 seconds.
sintesis.m: Learning took 11.5547 seconds.

```

- Agora para um video de 213.4KB de 93 frames com parametro $\tau = 5$ obtemos

```

estado.m: Learning took 5.3284 seconds.
modelo.m: Learning took 0.012143 seconds.
modelo.m: Learning took 0.023422 seconds.
matriz.m: Learning took 0.001219 seconds.
sintesis.m: Learning took 323.1624 seconds.

```

Referências

- [1] L. Ljung. *System Identification Theory for the User(2nd Edition)*. Prentice Hall, 1999.
- [2] Lu Yuan, Fang Wen, Ce Liu, and Heung-Yeung Shum, *Synthesizing Dynamic Texture with Closed-Loop Linear Dynamic System*, ECCV 2004.
- [3] A. Schodl, R. Szeliski, D. H. Salesin and I. Essa. *Video Textures*. In Proceedings of Siggraph00, pp. 489-498, 2000.
- [4] S. Soatto, G. Doretto and Y. N.Wu. *Dynamic textures*. In Proceedings of ICCV01, vol. 2, pp. 439-446, 2001.