Dynamic texture modeling and synthesis using Multi-kernel Gaussian process Dynamic model

MAKSIM KOPTELOV

University of Jean Monnet

Master 1 MLDM 2015/2016

Academic Supervisors:

Assoc. prof. Marianne CLAUSEL

Assoc. prof. Rémi EMONET

Prof. Olivier ALATA

DYNAMIC TEXTURES

What is a definition of texture?

- For a single image

  Realization from stationary stochastic process



What is a definition of texture?

- For a single image

  Realization from stationary stochastic process

- For a sequence of images

  Stochastic process of interest defined over space and time.

Why is it important?

Applications in Computer vision:

- video surveillance
- object tracking
- etc.

Application in video processing fields:

- video indexing
- video animation
- etc.

What we are doing

Input – video t1 seconds

Output – new videos t2 seconds, t2 > 0

- Dynamic texture can be captured from a video
- It can be used then to synthesize new videos with necessary length of time

Existing dynamic textures modeling and synthesis methods:

- *physics based* – costly, not universal
- *sampling based* – memory demanding, mostly manual
- *learning based methods* – more universal, automatic

CONTRIBUTION

Contribution

First mission – understand the model

Reimplementation of the method in Python – understand things:

- Gaussian process
- MK-GPDM
- sequence of steps of learning algorithm
- functions which must be optimized
- functions gradients derivation
- dimensionality analysis

Learning based methods

- high dimensionality – curse of dimensionality

dimensionality reduction process must be applied - infers by Gaussian process

- most of dynamic textures are not linear – learning algorithm cannot be linear

more flexible model must be found – based of Gaussian process

- big computational cost of optimization – necessary to have a good performance

first-order Markov model - based on Gaussian process

Why Gaussian process? Not many approaches can be used for modeling and synthesis

GAUSSIAN PROCESS

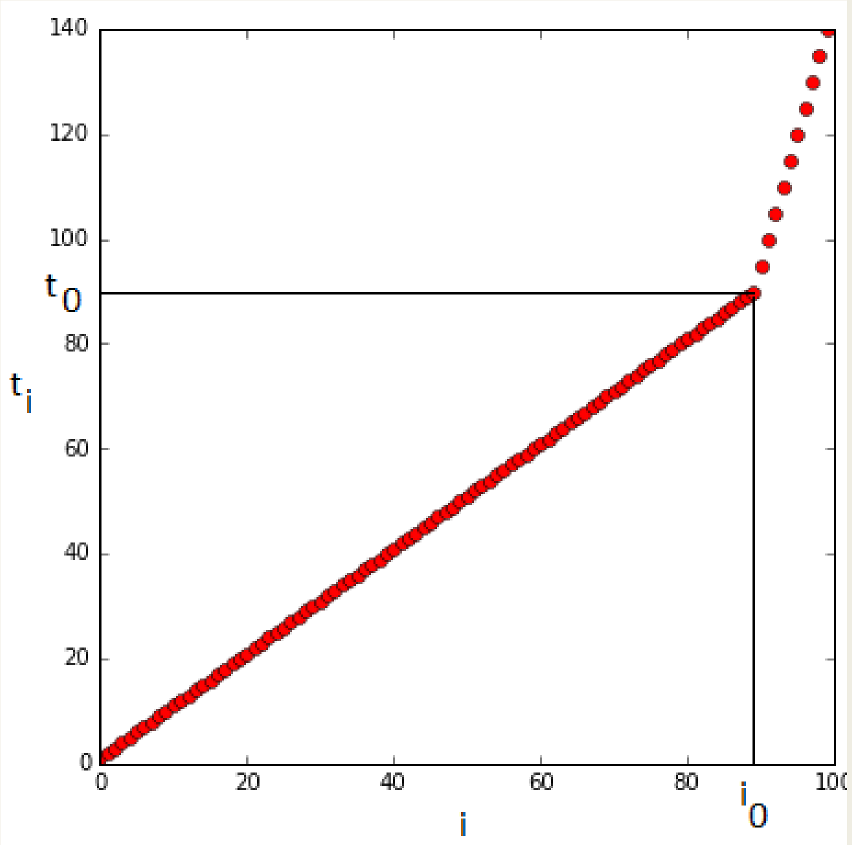
Gaussian process

Definition

Gaussian processes – infinite-dimensional generalization of multivariate normal distributions

Initially we have:

- d random variables (X_{t1},…, X_{td}) not equally spaced
- timeline t_i
- d = 100
- 100 time indexes t_i
- i_0 – index when space is increased
- not regular sampling

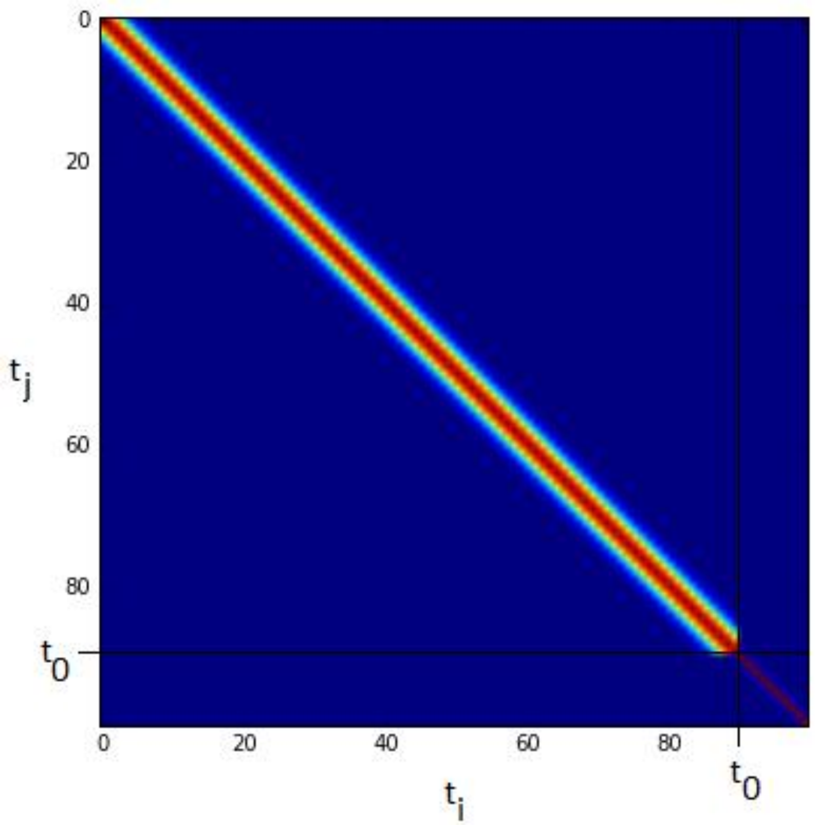


Gaussian process

Characterized by mean and covariance

- all the vector of interest are centered – zero mean initially
- assume a specific form for the covariance of any subsample (X_{t1},..., X_{td})

$$k(t_i, t_j) = cov(X_{t_i}, X_{t_j}) = \alpha \exp(-\frac{\|t_i - t_j\|^2}{2l^2})$$



Parameters:

a - scale parameter,

l - dispersion parameter

Gaussian process

From one covariance matrix and one sequence of time $(t_1,\dots,t_k)$ possible Gaussian distribution $(X_{t1},\dots, X_{tk})$ with zero mean and covariance C

On the axis:

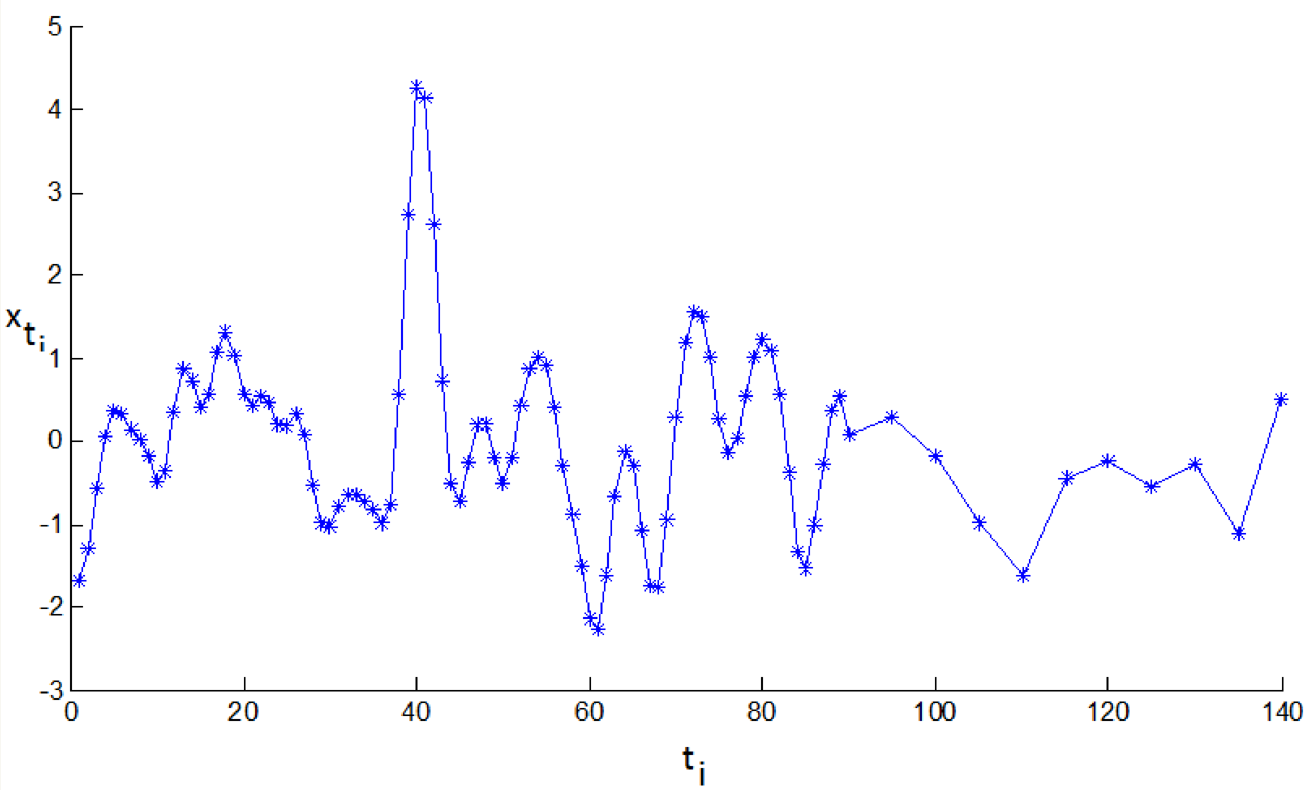
x_{ti} – values of random variables

t_i – time indexes

Trend of samples is not spoiled

by not equally spaced time indexes

and can be easily seen



Gaussian process

Now assume that we are conditioning our stochastic process for example:

$x_{t1} = 1$, $x_{t2} = 1$ with $t_1 = 40$, $t_2 = 80$

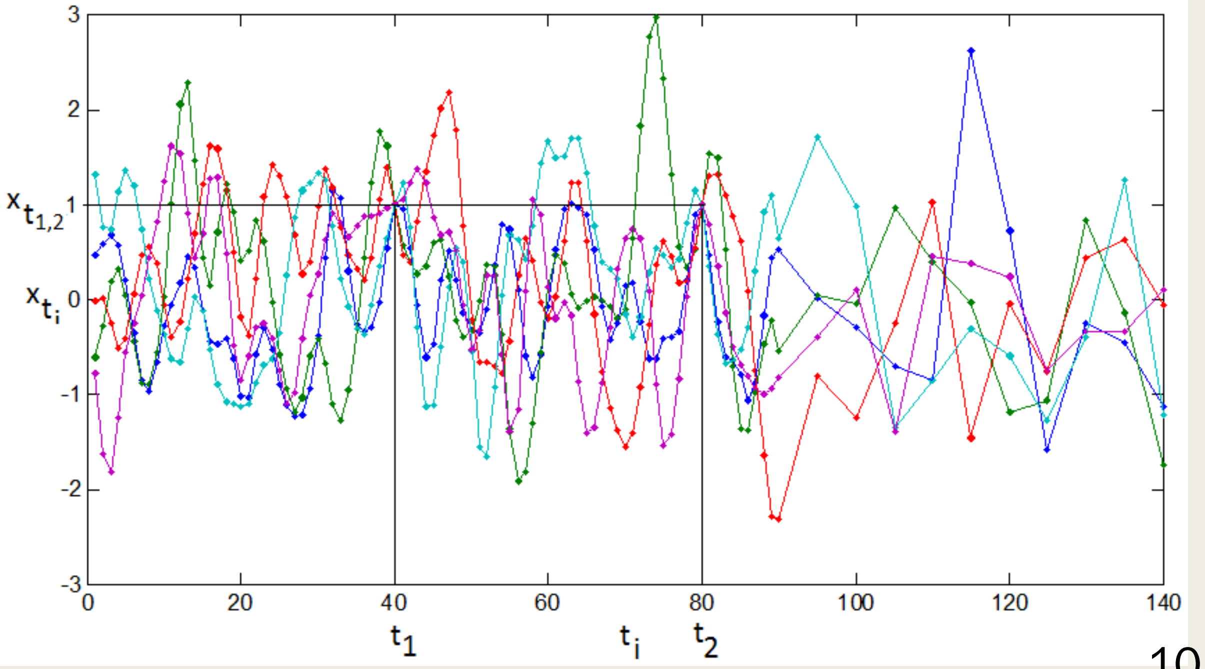
Compute new mean vector and covariance matrix (multivariate conditional density):

$$p(\mathbf{f}_*|\mathbf{f}) = \mathcal{N}(\mathbf{f}_*|\mu, \Sigma), \quad \mu = \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f}, \quad \Sigma = \mathbf{K}_{*,*} - \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}$$

f_* – Gaussian vector $x_* = x_{\{t \in T \setminus T'\}}$

f – Gaussian vector $x_f = x_{\{t \in T'\}}$

K – covariance matrices associated to two random variable sets $X_* = X_{\{t \in T \setminus T'\}}$, $X_f = X_{\{t \in T'\}}$



New set of realizations of multivariate conditional distribution:

Important property of Gaussian processes

Interpolation can be performed using this approach

GP with latent variables

- new class of models
- mappings from a latent space X ∈ R^Q to an observed space Y ∈R^D ,Q << D
- through a set of parameters W
- one kernel function is used

$$K_Y = k_Y(x_i, x_j) = \sum_{l=1}^{M} w_l k_l(x_i, x_j) + w_\delta \delta_{x_i, x_j}$$

- X, Y – multivariate Gaussian processes
- In a simplest case – probabilistic version of PCA
- likelihood of the full data set is given by:
- W can then be found through maximizing:

$$p(\mathbf{Y}|\mathbf{W}, \beta) = \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{W}, \beta)$$

Dynamic system

In this work dynamical texture modeling consists of two steps:

- dimensionality reduction
- dynamic texture learning

They can be expressed as:

$$x_{t+1} = f(x_t, A) + n_{x,t}$$ – dynamic model

$$y_t = g(x_t, B) + n_{y,t}$$ – dimensionality reduction

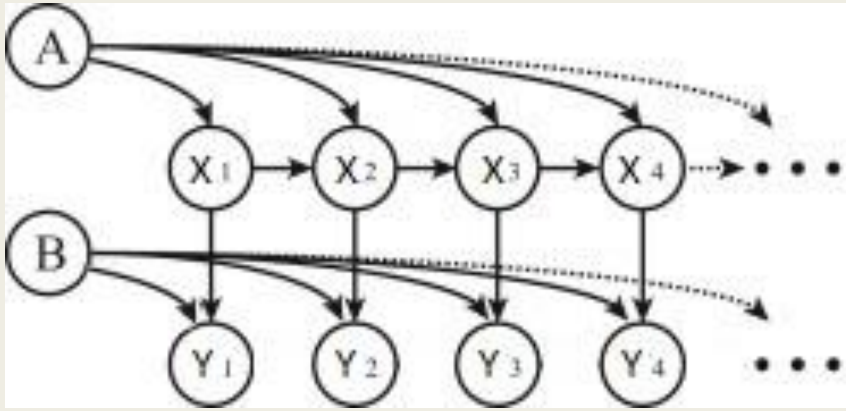
x_t - latent variable which affects dynamic behavior, $x_t \in R^Q$

y_t - column vector unfolded from the frame at time t, $y_t \in R^D$, D – large, Q<<D

$n_{x,t}$, $n_{y,t}$ - represent the noise

GP dynamic model

- latent variable model
- 2 mappings: from a latent space X to the observation space Y and dynamic behavior of latent variables
- 2 kernel functions
- X – sequence of row-vectors of artificial frames, Y – sequence of row-vectors of observed frames
- A – K_Y parameters θ, B – K_X kernel weights W



Y_i is a multivariate Gaussian process indexed by X_i expressed by likelihood

$$p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}) = \prod_{t=1}^{N} p(y_t|x_t,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{DN/2}|\boldsymbol{K}_Y|^{D/2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_Y^{-1}\boldsymbol{YY}^T\right)\right)$$

Multi-kernel GP dynamic model[1]

The likelihood extends to the following equation:

$$p(\boldsymbol{X}|\lambda,\boldsymbol{W}) = p(x_1)\prod_{t=2}^{N} p(x_t|x_{t-1},\lambda,\boldsymbol{W}) = p(x_1)\frac{1}{(2\pi)^{Q(N-1)/2}|\boldsymbol{K}_X|^{Q/2}}\exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{K}_X^{-1}\boldsymbol{X}_{2:N}\boldsymbol{X}_{2:N}^T\right)\right)$$

X – latent variable vector

Y – observed dynamic texture sequence vector

N – number of frames in original sample

Q – latent dimensionality

W – vector of weights of kernel functions $K = k_l$, $l \in [1,M]$

M – number of different kernel functions used

Reference [1]:
Z. Zhu, et al., Dynamic texture modeling and synthesis using multi-kernel Gaussian process dynamic model, Signal Processing (2015)
http://dx.doi.org/10.1016/j.sigpro.2015.10.025

Multi-kernel GP dynamic model[1]

To achieve nonlinear mapping from a latent space X to the observation space Y

special squared exponential covariance function is used

$$K_Y = k_Y(x_i, x_j) = \theta_1 \exp\left(-\frac{\theta_2}{2}(x_i - x_j)(x_i - x_j)^T\right) + \theta_3 \delta_{x_i, x_j}$$

To map dynamic behavior of latent variables

- Latent dynamic behavior varies greatly among different types of dynamic textures
- Difficult to design the most suitable kernel for a dynamic texture empirically

Multi-kernel dynamic model for dynamic texture modeling is proposed

$$K_X = k_X(x_i, x_j) = \sum_{l=1}^{M} w_l k_l(x_i, x_j) + w_\delta \delta_{x_i, x_j}$$

Reference [1]:

Z. Zhu, et al., Dynamic texture modeling and synthesis using multi-kernel Gaussian process dynamic model, Signal Processing (2015)
http://dx.doi.org/10.1016/j.sigpro.2015.10.025

Algorithm understanding

Maximum a posteriori estimation – main method. In Matlab specific library for optimization is used.

In general two main steps:

- fix W and perform optimization with respect to X, θ and λ using SCG (Scaled conjugate gradient)

$$F(X,\theta,\lambda) = -lnP(X,\theta,\lambda|Y) = \frac{D}{2}ln|K_Y| + \frac{1}{2}tr(K_Y^{-1}YY^{-1}) + \frac{Q}{2}ln|K_X| + \frac{1}{2}tr(K_X^{-1}X_{2:N}X_{2:N}^T) + \sum_i \theta_i + \sum_{i,j}(\lambda_i)_j + C$$

  W – weights vector

  X – latent variable

  θ – vector of hyperparameters of kernel K_Y

  Λ – vector of parameters of kernel K_X

- fix obtained X, θ and λ and perform optimization with respect to W using gradient descent

$$F(W) = \frac{Q}{2}ln|K_X^{-1}| + \frac{1}{2}tr(K_X^{-1}X_{2:N}X_{2:N}^T) + \alpha\|W\|_2$$

Functions are not convex – repeat these two main steps for I times, I – fixed

RESULTS

Matlab implementation properties

Implementation given in Matlab has problems:

- Inexplicable instability – often crashes with SVD computation error
- Static result – sometimes generated dynamic texture does not move at all
- Repetition of original sequences of frames

Attempt	Linear	RBF	Poly	RatQuad	MLP	Matern32
1	0.040278	0.682626	0	0	0	0.277096
3	0.067891	0.720069	0	0	0.212038	0
7	0.003638	0.368365	0	0.345235	0.069727	0.213032

Python implementation properties

Implementation in Python:

- GPLVM does not have specific kernel function to capture the dynamic behavior
- Temporal structure is included in covariance

The results are quite surprising:

- it is stable
- it is able to generate new sequences of dynamic textures without visible repetitions
- it takes a lot of time to perform optimization
- visible result is still not good due to some random noise

EXAMPLE

Examples

Matlab implementation:

- Input videos ***sunshade.avi*** and ***straw.avi***
- Example of a good result (sunshade.avi)
- Example of a bad result (straw.avi)

Python implementation

- Example of a good result (straw.avi)

PERSPECTIVES

Perspectives

- MK-GPLVM reimplementation in Python is not finished – problems with gradients
- no numerical criteria for quality measuring – implement an evaluation method
- GPLVM implementation in Python is done and provides quite interesting result – improve the results
- try wavelets to reduce dimensionality of original input to decrease time for optimization

THANK YOU !