Image Quality Assessment based on Machine Learning for the Special Case of Computer-generated Images

André Bigand (PhD, HDR) (bigand@lisic.univ-littoral.fr)

www-lisic.univ-littoral.fr/~bigand/Tutorial

Visigrapp 01/03/2017
Room Aregos
Schedule

1. Introduction

2. Natural scenes quality assessment

3. Computer-generated images

4. Neural Networks, RBF and MLP

5. SVM (Support Vector Machine), RVM (Relevance Vector machine)

6. Other methods, extension of SVM

7. Some results

8. Conclusion
Introduction

Image quality is a well-known paradigm:

• Natural scenes: reflected light (Bovik, Beghdadi, …)

• Computer-generated images (cartoons, virtual reality, surgery, …): emitted light

• The both are now often considered!
Natural scenes quality assessment

• How to recognize a nice picture?
• What is image quality?
• Light intensity, patterns, shapes, noise, saliency....
• Art
Human Visual System (HVS): Modelization

- Biological model (RVB for colors, …)
- Psychological model (saliency)
- Cognitive model (shapes, …)
What about art?

Why do I prefer Kandinsky: subjective aspect....
Extracting visual features for classifying paintings

- Style (Cubist,...), genre ((Portrait, landscape, ...), Artist

- Use of Support Vectors Machines (SVMs)

- Painting : vertex of a graph

- Wij adjacency matrix

- 64k paintings of Wikiart dataset
Measuring Creativity scores according to dates (from 1400 to 2000)
Image quality models

- Natural scenes statistics: average, variance, …, entropy (aesthetic model), … computed from error ($I_{ref} - I_{proc}$)
- Human Visual System (HVS): very complex! Perceptual aspects:
  - contrast,
  - visual masking (target vs mask),
  - spatial frequency,
- Perceptual image segmentation (color, texture, shape, intensity, scale, …): features
Image quality models

- Numerous applications:
  - Media-layer objective video quality assessment (LIVE video quality database)
  - Images coding (JPEG, MPEG, ...)
- Objective quality metrics: scores
- Subjective quality evaluation: group of users (HVS): reliable but time consuming!

Image Quality Assessment: a difficult task!

The use of Machine Learning? (Lahoudou, Beghdadi: 2010)
How to measure image quality (score)?

(Feature and Dimensionality)
Machine Learning: The curse of dimensionality (R.Bellman)

- Data represented by a vector $v \in [0, 1]^D$, $D$ dimension
- $D=1$ : with 100 points interval between points is $10^{-2}$
- $D=10$ : for the same separation interval, $10^{20}$ points are needed!

- Number of points (supervised learning) is often limited
Computer-generated images

- Light sources (direct or indirect)
- Reflectance functions
- Image built on a virtual camera (point of view)

- What is a nice picture?
- Perceptual noise (lack of information)
Image Generation Techniques

- Numerous rendering algorithms have been proposed since the 70's: ray-tracing [Whitted 1979], radiosity [1984], path-tracing [Kajiya 86], REYES [1987].
- Path-tracing: **random walk** to solve the rendering equation
- **Global illumination** (a) A scene can be decomposed in direct lighting (b) and indirect lighting (c). The direct lighting is the light coming from a source to a point and reflected toward the camera. The indirect lighting is the light coming from a source and reflected several times before reaching the camera.
Global Illumination

a

b

c
Path Tracing Method

The global illumination algorithm generates stochastic paths from the camera to the 3D scene. For each intersection of a path with the surface a direction of reflection or refraction is randomly extracted.
Monte Carlo Method

For each pixel, the final luminance is the average of the function of light for all generated paths:

$$I = \frac{1}{N} \sum_{i=1}^{N} L(x_i)$$

$$L(x) = L_e(x) + L_r(x)$$

$L_r(x)$: The function of reflected light.

$L_e(x, w)$: The function of emission light.
Method's Parameters:
The Cornell box scene (diffuse materials + one light source) is a classic scene for testing rendering with global illumination and color blending. Rendered with:
max path length = 1, 2, 8, number of paths per pixel = 1024
Then from a noisy image we can obtain a reference image which is not affected by noise after a certain number of paths.

1 path/pixel (noisy scene) 10 paths/pixel Reduce noise 100 paths/pixel Reduce noise 10100 paths/pixel (reference scene)
Objective

Stopping Criterion in Case of Producing Synthetic Images in Global Illumination Algorithms

Reference Quality Assessment

Non-Reference or Reduced Reference Quality Assessment

RBF, SVM, RVM

SNN, ...

Quality Measure

MSE (Mean square error)
SNR (Signal to noise Ratio)
PSNR (Peak Signal to noise Ratio)
VDP (Visible Difference Predictor)

Performance

Comparison between NNs
Comparison between learning models and human vision system (NN: Neural Network)
Data Acquisition

Each image of 512x 512 pixels is virtually cut into 16 sub-images of 128x 128 pixels because:

a) The evaluation methods should work on a small set of data.
b) The noise thresholds are different for each location in the image.
The observer points the areas where differences are perceived between the current image and the reference one.

Each operation causes the display of the next level sub-image by reducing visually the noise in this image’s subpart.
Thresholds Computing
(Experimental benchmark)

The results were recorded for 33 different observers and the average number of paths that are required for each sub-image to be perceived as identical to the reference one was computed.

<table>
<thead>
<tr>
<th>Sub-images</th>
<th>Thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4579</td>
</tr>
<tr>
<td></td>
<td>6631</td>
</tr>
<tr>
<td></td>
<td>5818</td>
</tr>
<tr>
<td></td>
<td>5906</td>
</tr>
<tr>
<td></td>
<td>3271</td>
</tr>
<tr>
<td></td>
<td>2222</td>
</tr>
<tr>
<td></td>
<td>2261</td>
</tr>
<tr>
<td></td>
<td>2009</td>
</tr>
<tr>
<td></td>
<td>2400</td>
</tr>
<tr>
<td></td>
<td>2324</td>
</tr>
<tr>
<td></td>
<td>2098</td>
</tr>
<tr>
<td></td>
<td>2735</td>
</tr>
<tr>
<td></td>
<td>2955</td>
</tr>
<tr>
<td></td>
<td>2344</td>
</tr>
<tr>
<td></td>
<td>2002</td>
</tr>
<tr>
<td></td>
<td>2190</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub-images</th>
<th>Thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1866</td>
</tr>
<tr>
<td></td>
<td>1694</td>
</tr>
<tr>
<td></td>
<td>1784</td>
</tr>
<tr>
<td></td>
<td>1978</td>
</tr>
<tr>
<td></td>
<td>3302</td>
</tr>
<tr>
<td></td>
<td>3465</td>
</tr>
<tr>
<td></td>
<td>4699</td>
</tr>
<tr>
<td></td>
<td>2827</td>
</tr>
<tr>
<td></td>
<td>2134</td>
</tr>
<tr>
<td></td>
<td>1442</td>
</tr>
<tr>
<td></td>
<td>1493</td>
</tr>
<tr>
<td></td>
<td>1683</td>
</tr>
<tr>
<td></td>
<td>1989</td>
</tr>
<tr>
<td></td>
<td>1934</td>
</tr>
<tr>
<td></td>
<td>1816</td>
</tr>
<tr>
<td></td>
<td>1975</td>
</tr>
</tbody>
</table>
General Methods

Stopping criterion for computer-generated images

Reference Image

Distances measures
- SNR (Signal to noise Ratio)
- PSNR (Peak Signal to noise Ratio)
- VDP (Visual Difference Predictor)
- SSIM

Unknown Reference Image

MLP, RBF, SVM, RVM

MOS

Performance
- Bayesian learning (good performance).
- Comparison between RBF, SVM, MLP, RVM and MOS.

(MOS : Mean Opinion Score)
Reference Image known $I(i,j)$

The SNR is defined using:

$$SNR = 10 \log_{10} \left( \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{1}{NM} (\widetilde{I}(i, j))^2 \right)$$

With average quadratic error $MSE$ by:

$$MSE = \frac{1}{NM} \left\{ \sum_{i=1}^{M} \sum_{j=1}^{N} |I(i, j) - \widetilde{I}(i, j)|^2 \right\}$$

And for ending PSNR defined using:

$$PSNR = 10 \log_{10} \left( \frac{\text{Valeur Max du signal}}{MSE} \right)^2$$
Another error detection method is VDP method (Visual Difference Predictor):

Reference Image known

1. Image1
   - Retinian Response
   - Contrast Sensibility
     - Pyramidal decomposition (frequency and orientation)
     - Mask
     - Distance
     - Error Detection Probability

2. Image2
   - Mask
Structural Similarity (SSIM) index

- SSIM index is a method for measuring the similarity between two images.
- The SSIM index is a full reference metric; in other words, the measuring of image quality based on an initial uncompressed or distortion-free image as reference. SSIM is designed to improve on traditional methods like peak signal-to-noise ratio (PSNR) and mean squared error (MSE), which have proven to be inconsistent with human eye perception.

- The difference with respect to other techniques mentioned previously such as MSE or PSNR is that these approaches estimate perceived errors; on the other hand, SSIM considers image degradation as perceived change in structural information. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the objects in the visual scene.
Structural Similarity (SSIM) index

The SSIM metric is calculated on various windows of an image. The measure between two windows $x$ and $y$ of common size $N \times N$ is:

$$SSIM(x,y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

- $\mu_x$ the average of $x$;
- $\mu_y$ the average of $y$;
- $\sigma_x^2$ the variance of $x$;
- $\sigma_y^2$ the variance of $y$;
- $\sigma_{xy}$ the covariance of $x$ and $y$;
- $c_1 = (k_1L)^2$, $c_2 = (k_2L)^2$ two variables to stabilize the division with weak denominator;
- $L$ the dynamic range of the pixel-values (typically this is $2^{\#\text{bits per pixel}} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.
Due to non-uniformity of human visual system to perceive color differences without distinguishing color information from luminance information, $L^*a^*b^*$ color space is used for computer-image generation.

- $L^*$: luminance information.
- $a^*b^*$: chrominance information. (color oppositions: green and red, yellow and blue)

Luminance information is used as input information, since perceptual noise is brought by luminance.
Neural Networks and Image quality

- Neural Networks (NNs) are *structures* that tend to mimic human neural activity
- They are often linked with supervised learning (they need examples to adjust their parameters)
- Once their parameters are adjusted, NNs are very efficient to take decision!
- They take image *dimensionality* problem into account.
Machine Learning Paradigms

- Supervised-learning (requires a trainer)
- Unsupervised learning, semi-supervised learning, (reinforcement learning)
- NNs generalization: cross-validation (k-fold,...)
Neural networks for vision

- NNs used for Object Recognition Systems:
Feature Selection (1)

- Towards ultra-high dimensional data (video, 4k)
- Low-level features, Feature selection process:
Feature Selection (2)

- Computer-generated images: features and perceptual noise!
- Our work is based on **Noise features** for Image Quality Evaluation:
  - $L(i,j) = 'L' \text{ component of Lab image}$
  - $L_d(i,j) = \text{denoised version of } L(i,j)$
  - $e(i,j) = |L(i,j) - L_d(i,j)|$
- Noise statistics
Noise Features

We apply to L component four different denoising algorithms: Linear filtering with averaging filters of sizes 3x3 and 5 x 5, linear filtering with Gaussian filters of same sizes and with standard deviations in \{0.5,1,1.5\}, median filters and adaptive Wiener filters of same sizes. Low pass filtering using averaging or Gaussian filters helps to model the high frequency noise, non-linear median filtering addresses the salt and pepper noise, and Wiener filtering which can tailor noise removal to the local pixel variance. Next, the obtained image is denoised via wavelet analysis. After using the one stage 2-D wavelet decomposition, an input image is decomposed into four sub-bands namely low-low (LL), low-high (LH), high-low (HL) and high-high (HH) sub-bands.
Noise Features

- Wavelet Analysis:
Noise Features

• The mean and the standard deviation are extracted from the 13\textsuperscript{th} obtained errors:

\[
F^{(1)} = \frac{1}{MN} \sum \sum e(i,j)
\]

\[
F^{(2)} = \left( \frac{1}{MN} \sum \sum (e(i,j) - F^{(1)}(L))^2 \right)^{0.5}
\]

_ MN is the dimension of the image matrix 'L'_

• **Feature Vector** = 26 features to characterize noise
Noise Features

- **Standard deviation variation per block**
- **Mean variation per block**

Graphs showing:
- Mean value of $f^{(2)} = 8.71$ for Block 3 (Minimum of high frequency noise)
- Mean value of $f^{(2)} = 2.37$ for Block 6
- Mean value of $f^{(1)} = 1.82$ for Block 11 (Maximum of high frequency noise)
- Mean value of $f^{(1)} = 4.23$ for Block 3 (Minimum of high frequency noise)

Number of sub-images = 101 per Block
NNs and Computer-generated Images quality?

The real potential of artificial intelligence comes when having machines learnt to solve problems without being told how to solve them. This is the case in the area of computer-generated image. In this tutorial, we provide some results and some ideas about the use of NNs. We will also show some results about realistic images at present.
System block Diagram

Add K = 100 new paths

G. I renderer

Noise Quality Indexes

noise perception model

Filtering

Extract Subimage

Scene data

end

no

noise left?

yes

yes

no
RBF and MLP Neural Network

The RBF neural network is composed of:

• **Input Layer:** Each input variable is interpreted towards the hidden layer.

• **Hidden Layer:** Input space is non-linearly projected towards a new space using Gaussian functions and product of Gaussian functions is computed.

• **Output Layer:** This process makes it possible to deliver a « decision » or the result of what is expected.
Radial Basis Functions
Radial Basis Functions (Neuron Structure)
Equations of Radial Basis Functions

- **Hidden Layer:** The radial basis functions are defined by the formula:
  \[
  \phi_k(x, c_k) = \left( \prod_{j=1}^{D} K(x_j, c_{kj}) \right) / (\sigma^2)
  \]
  
  \[x = [x_1 \ x_2 \ \cdots \ x_D] \quad : \text{is the input vector}
  \]
  
  \[c = [c_{k1} \ c_{k2} \ \cdots \ c_{kD}] \quad : \text{are the Gauss centers}
  \]
  
  \[K(x_j, c_{kj}) = \exp\left( -(x_k - c_{kj})^2 \right)
  \]
  
  \[\sigma \quad : \text{is the kernel parameter and } D \text{ is the number of centers.}
  \]

- **Output Layer:**
  \[y = \sum_{k=1}^{M} w_k \Phi_k + w_0\]
RBF Network Learning

Parameters are computed using quadratic error minimization:

\[ E = \frac{1}{2} \sum_{i=1}^{s} [d_i - y_i]^2 \]

\( y_i \) : Actual output.

\( d_i \) : Desired output target.
Kernel Parameters Adjustment

Kernel centers and variances for iteration n to n-1:

\[ c_{ki}(n) = c_{ki}(n-1) - \alpha \frac{\partial E}{\partial c_{ki}} \]

with

\[ \frac{\partial E}{\partial c_{ki}} = -\frac{2}{\sigma_{ki}^2} \sum_{j=1}^{S} \Phi[x, c_k, \sigma_k][x_i - c_{ki}]w_{jk}[d_j - y_j] \]

\[ \sigma_{ki}(n) = \sigma_{ki}(n-1) - \beta \frac{\partial E}{\partial \sigma_{ki}} \]

with

\[ \frac{\partial E}{\partial \sigma_{ki}} = -\frac{2}{\sigma_{ki}^3} \sum_{j=1}^{S} \Phi[x, c_k, \sigma_k][x_i - c_{ki}]^2w_{jk}[d_j - y_j] \]
Weights updating

Weights updating for the output layer is given using:

\[ W(n) = W(n-1) - F \frac{\partial E}{\partial W} \]

with \[ \frac{\partial E}{\partial W} = \phi [d_n - y_n] \]
MLP Network

- MLP network has same structure as RBF network (3 layers).

- If output layer function is the same thus the output is given by:

\[
\hat{y} = \hat{W}^T \sigma(\hat{V}^T x)
\]

- Network parameters are computed using the well-known back-propagation algorithm:

\[
W(n) = W(n-1) - F \frac{\partial E}{\partial W}
\]

\[
V(n) = V(n-1) - G \frac{\partial E}{\partial W}
\]
MLP Network

- Weights are updated using error and following equations are obtained:

\[ W(n) = W(n-1) + F\hat{\sigma} \ (d-y)^T \]

\[ V(n) = V(n-1) + Gx(\hat{\sigma}'^T \ \hat{W} \ (d-y))^T \]
Over-fitting

• Error minimization using back-propagation algorithm often produces over-fitting.
The learning technique SVM has the objective to find the optimal separating hyperplane which:

Minimize: \( \frac{1}{2} \|w\|^2 \)

With the constraint: \( \exists w \in \mathbb{R}^n, w_0 \in \mathbb{R}: y_i (\langle w, x \rangle + w_0) \geq 1, \forall i \in 1 \cdots n \)
SVMs and MLPs
The hyperplane that optimally separates the data is the one that:

Minimize: $\frac{1}{2}\|w\|^2$

With the constraint: $\exists w \in \mathbb{R}^n, w_0 \in \mathbb{R}: y_i(<w,x>+w_0) \geq \pm 1, \forall i \in 1 \cdots n$

This problem consists to minimize the following Lagrangian with respect to variables $w$, $w_0$ and to maximize with respect to $\alpha_i$.

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(<w,x>+w_0)-1)$$
SVM Learning

- This problem can be written using the following dual form:

\[
\text{Maximize: } \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j <x_i,x_j> \\
\text{With the constraint: } \exists \alpha_i \in \mathbb{R}, \alpha_i \geq 0: \sum_{i=1}^{n} \alpha_i y_i = 0, \forall i \in 1 \cdots n
\]

Where \( \alpha_i \) are the Lagrange multipliers.

- During the optimization process, the Lagrange multipliers are nulls for all points different from supports vectors.
SVM Learning

- Instead of using a hard separator, a constraint can be added to the optimization problem. The following term should be minimized:

\[
\frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i
\]

With C is a given constant and \( \xi \) is a penalty factor

- This problem can then be solved by using a radial basis function which verifies the Mercer condition:

\[
\phi(x_i, x_j) = e^{-\sigma \| x_i - x_j \|^2}, \sigma > 0
\]
Experimental Results

- The network parameters are optimized using the V-times Cross-Validation technique.

<table>
<thead>
<tr>
<th>C(SVM)</th>
<th>Avg. Numbers of Support vectors</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>724.44</td>
<td>95.42</td>
</tr>
<tr>
<td>2</td>
<td>712.18</td>
<td>95.54</td>
</tr>
<tr>
<td>4</td>
<td>708.24</td>
<td>96.03</td>
</tr>
<tr>
<td>8</td>
<td>708.13</td>
<td>96.03</td>
</tr>
<tr>
<td>16</td>
<td>707.59</td>
<td>96.00</td>
</tr>
<tr>
<td>32</td>
<td>707.59</td>
<td>96.03</td>
</tr>
</tbody>
</table>

Average numbers of support vectors and precision for different values of the parameter C
Experimental Results
(Number of parameters: 26x258= 6708)

- The output of the network is '-1' if the image is considered noisy or '+1' if the image is considered non-affected with noise.
function c_learninga
load threshold_lr
X = [];
D = [];
j = 1;
str = 'bar2';
for i = 1:16
    if (i == 16)
        X1  = zeros(101,26);
        D1  = zeros(101,1);
        c  = 1;
        for e = 100:100:10100
            s  = sprintf('c:\Users\toshiba\Desktop\images\%s\Zone%d\image%d.png', str, i-1, e);
            I  = imread(s);
            cform = makecform('srgb2lab');
            L = applycform(I,cform);
            L = afiltera(L(:,:,1));
            th = thr(i,j);
            if e < th
                X1(c,:) = L(:);
                D1(c,:) = -1;
            else
                X1(c,:) = L(:);
                D1(c,:) = 1;
            end
        c = c+1;
    end
ln  = size(X1,1);
lc  = size(X1,2);
XD  = X1(ln,:);
for k = 1:ln
    X1(k,:) = X1(k,:)- XD;
end
X   = [X; X1];
D   = [D;D1];
end

SVM method: algorithm (1)
function svmlearninga;

    clear all
    close all
    tstart = tic;
    E  = [];
    Er  = [];

    load learningf

%*******************************************************************************
% Training
%*******************************************************************************
strain=svmtrain(X,D,'kernel_function','rbf','rbf_sigma',4,'boxconstraint',2^6);
telapsed = toc(tstart)
y = svmclassify(strain,X);
ln = size(X,1)
for i = 1:ln
    if abs(y(i)-0) < abs(y(i)-1)
        R(i) = 0;
    else
        R(i) = 1;
    end
end
%*******************************************************************************
% MSE Learning Base
%*******************************************************************************
nrm    =  zeros(101,1);
SVM method: algorithm (3)

DEMO

function

svmlearninga
Learning base Images (Image number: 33, Type: Noisy Images)
Learning base Images (Images number: 27, Type: Unnoisy Images)
Test base Images (Images number: 33, Type: Noisy Images)
Test base Images (Image number: 27, Type: Unnoisy Images)


No: 7 (R: 8000)  No: 8 (R: 8200)  No: 9 (R: 8400)  No:10 (R: 8600)  No:11 (R: 8800)  No:12 (R: 9000)


No:19 (R: 10400)  No:20 (R: 10600)  No:21 (R: 10800)  No:22 (R: 11000)  No:23 (R: 11200)  No:24 (R: 11400)

No:25 (R: 11600)  No:26 (R: 11800)  No:27 (R: 12000)
From SVM to RVM

- What is the problem?
- The Relevance Vector Machine (RVM) is linked with Bayes' theory.
- The SVMs have a huge capability for regression and classification (the support vectors are viewed as kernel centers).
RVM (Relevance Vector Machine) vs. SVM

1. RVM takes some advantages over SVM in particular in the probabilistic aspect to detect the uncertainties during prediction.

2. In SVM resolution, the number of supports vectors will increase with the size of the learning set, the method RVM uses a small number of relevance vectors independently of the size of the learning set.

3. The approach SVM requires to use radial basis functions which verify the Mercer theorem, which is not the case in the approach RVM.

4. In SVM, It is necessary to estimate the error by finding the value of the constant C which requires necessary a cross validation technique.
RVM Method

• The method RVM has the objective to prune the nuisance parameters.

• The parameter that does not maximize the marginal probability function is pruned in order to minimize system complexity.

• The likelihood of the data set is defined as:

\[
p(t \mid w, \sigma^2) = (2\prod \sigma^2)^{-N/2} \exp\left\{-\frac{1}{2\sigma^2} \| t - \phi w \|^2 \right\}
\]

• The maximization of this function is similar to the back-propagation method and will lead to over-learning.

RVM Method

To solve this problem, the likelihood function is complemented by a prior over the parameters:

\[ p(w|\alpha) = (2\pi)^{-M/2} \prod_{m=1}^{M} \alpha_m^{1/2} \exp\left(-\frac{\alpha_m w_m^2}{2}\right) \]

- This distribution gives a good approximation for the data of the learning set.
To solve this problem, the likelihood function is complemented by a prior over the parameters:

\[ p(w|\alpha) = (2\pi)^{-M/2} \prod_{m=1}^{M} \alpha_m^{1/2} \exp\left(-\frac{\alpha_m w_m^2}{2}\right) \]

- This distribution gives a good approximation for the data of the learning set.
The posterior distribution over the weights is calculated by applying the Bayes rule:

\[
p(w|t,\alpha,\sigma^2) = \frac{\text{likelihood} \times \text{prior}}{\text{normalising Factor}}
\]

\[
p(w|t,\alpha,\sigma^2) = \frac{p(d|w,\sigma^2)p(w|\alpha)}{p(d|\alpha,\sigma^2)}
\]

\[
= (2\pi)^{-\frac{(N+1)}{2}} |\Sigma|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2} (w-\mu)^T \Sigma^{-1} (w-\mu) \right\}
\]
In this case, the posterior mean and the covariance are Gaussian. They are defined as:

\[ \mu = \sigma^{-2} \sum \phi^T t \]
\[ \Sigma = \left( \sigma^{-2} \phi^T \phi + A \right)^{-1} \]

\[ A = \text{diag}(\alpha_1, \alpha_2 \cdots \alpha_M). \]

The main objective of Bayesian method is the marginalization process which prunes the nuisance parameters.
FRVM (Fast Relevance Vector Machine)

- The FRVM algorithm adds basis functions to an empty model in order to increase the marginal likelihood. The term $C$ can be decomposed as:

$$C = C_{-i} + \alpha_i^{-1} \phi_i \phi_i^T$$

- The logarithm of the marginal likelihood can be written as:

$$\ell(\alpha) = \ell(\alpha_{-i}) + \frac{1}{2} \left[ \log \alpha_i - \log (\alpha_i + s_i) + \frac{q_i^2}{\alpha_i + s_i} \right]$$

Where $s_i = \phi_i^T C_{-i}^{-1} \phi_i$ and $q_i = \phi_i^T C_{-i}^{-1} t$. These values can be calculated based on Woodbury.

- The logarithm has a unique maximum with respect to $\alpha_i$:

$$\alpha_i = \frac{s_i^2}{q_i^2 - s_i} \quad \text{if} \quad q_i^2 > s_i \quad \alpha_i = \infty \quad \text{if} \quad q_i^2 \leq s_i$$
FRVM Learning Algorithm (Parameters initialization)

- Initialize $\sigma^2$ to some sensible value.
- Initialize with a single basis vector $\phi_i$
- Initialize:
  \[
  \alpha_i = \frac{\|\phi_i\|^2}{\left(\|\phi_i^T t\|^2 / \|\phi_i\|^2\right) - \sigma^2}
  \]
  all other $\alpha_m$ are set to infinity.
- Compute $\Sigma$ and $\mu$ (which are scalars initially), along with initial values of $s_m$ and $q_m$ for all $M$ bases.
FRVM Learning Algorithm

Select a candidate basis vector $\phi_i$ from the set of all $M$

Compute $\theta_i = q_i^2 - s_i$

If $\theta_i > 0$ and $\alpha_i < \infty$ ($\phi_i$ is in the model) re-estimate $\alpha_i$

If $\theta_i > 0$ and $\alpha_i = \infty$ add $\phi_i$ to the model with updated $\alpha_i$

If $\theta_i \leq 0$ and $\alpha_i < \infty$ delete $\phi_i$ from the model and set $\alpha_i = \infty$

Update: $\sigma^2 = ||t - y||^2 / (N - M + \sum_m \alpha_m \Sigma_{mm})$

Recompute $\Sigma$ and $\mu$ and all $sm$ and $qm$

The optimization condition is satisfied?

End

No

Yes
Experimental Results

- The scene named "Bar" is used for learning and the scene "Class" is used for testing.

- The input of the network is a vector of 128x128 components which is obtained by pixel-wise subtraction between the original image and the de-noised one.

- The difference of two sub-images is provided to the model: A sub-image called the reference one which is quickly ray traced image of scenes and one of the test sub-images.
Experimental Results

- The scene named "Bar" is used for learning and the scene "Class" is used for testing.

- The input of the network is a vector of 128x128 components which is obtained by pixel-wise subtraction between the original image and the de-noised one.

- The difference of two sub-images is provided to the model: A sub-image called the reference one which is quickly ray traced image of scenes and one of the test sub-images.
Experimental Results

(a) Variation of the learning precision with respect to standard deviation.

(b) Variation of the average numbers of vectors or relevance vectors with respect to standard deviation.
Choose images for the experimental part of the process

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4579</td>
<td>6631</td>
<td>5818</td>
<td>5906</td>
</tr>
<tr>
<td>3271</td>
<td>2222</td>
<td>2261</td>
<td>2009</td>
</tr>
<tr>
<td>2400</td>
<td>2324</td>
<td>2098</td>
<td>2735</td>
</tr>
<tr>
<td>2955</td>
<td>2344</td>
<td>2002</td>
<td>2190</td>
</tr>
</tbody>
</table>
Experimental Results

The thresholds obtained by the HVS for the testing scene "Cube"
Experimental Results

(a) Variation of the average quadratic errors for each sub-image on the learning scene "Bar".
(b) Variation of the average quadratic errors for each sub-image on the testing scene "Class".
Experimental Results

(a) Variation of the average quadratic errors for each sub-image on the testing scene "Cube".

(b) Threshold variation for each sub-image on the testing scene "Cube."
Learning model structure

\[ D = 128 \times 128 \]
Learning method

- Cross-validation is used to optimize network parameters. Learning algorithm stops when average quadratic error (on the 60 images) does not change on test base. For each block error is defined using:

\[ E_{avg} = \frac{1}{N} E \text{ (N=60 Images)} \]

Total error for the 16 blocks is defined using:

\[ E_z = \frac{1}{z} E_{avg} \text{ (z=16 Zones)} \]
Kernel width optimization for the RVM method (Gauss Radius = 1250)

Average of Relevant vector numbers for the 16 blocks

Minimum Average = 19.0625
Kernel width optimization for the RVM method (Gauss Radius = 1250)

Minimum Total Error = 0.0046

Total error on the test base for the 16 blocks
Kernel width optimization for the SVM method (Gauss Radius = 300)

Average value of support vector number for the 16 blocks

Minimum Average = 24.93
Kernel width optimization for the SVM method (Gauss Radius = 300)

Minimum Total Error = 0.0031

Toral error on the test base for the 16 blocks.
**C parameter optimization for the SVM method (C= 512)**

<table>
<thead>
<tr>
<th>Valeur du paramètre C</th>
<th>Erreur totale sur la base de test</th>
<th>Moyenne des nombres des vecteurs supports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.042</td>
<td>56.87</td>
</tr>
<tr>
<td>2</td>
<td>0.022</td>
<td>53.18</td>
</tr>
<tr>
<td>4</td>
<td>0.0047</td>
<td>45.56</td>
</tr>
<tr>
<td>8</td>
<td>0.0042</td>
<td>39</td>
</tr>
<tr>
<td>16</td>
<td>0.0057</td>
<td>35.25</td>
</tr>
<tr>
<td>32</td>
<td>0.0047</td>
<td>31.31</td>
</tr>
<tr>
<td>64</td>
<td>0.0052</td>
<td>29.06</td>
</tr>
<tr>
<td>128</td>
<td>0.0047</td>
<td>27.25</td>
</tr>
<tr>
<td>256</td>
<td>0.0047</td>
<td>25.37</td>
</tr>
<tr>
<td><strong>512</strong></td>
<td><strong>0.0031</strong></td>
<td><strong>24.93</strong></td>
</tr>
<tr>
<td>1024</td>
<td>0.0036</td>
<td>24.93</td>
</tr>
</tbody>
</table>
Parameters of the hidden layer optimization (RBF or MLP)

- For each block, neural network is optimized using cross-validation.
- The right neuron number is given when average of quadratic error is minimized on the test base.
- Learning is made on the learning base and average error is tested (on the test base) at each iteration. Calculation is stopped when average error reaches a minimal value on the test base.
Neuron number optimization for the MLP
Learning parameters optimization for the MLP
Learning parameters optimization for the MLP
Neuron number optimization for the RBF network

- Zone 1: Minimum = 30
- Zone 2: Minimum = 40
- Zone 3: Minimum = 40
- Zone 4: Minimum = 40
- Zone 5: Minimum = 30
- Zone 6: Minimum = 40
- Zone 7: Minimum = 20
- Zone 8: Minimum = 40
- Zone 9: Minimum = 20
- Zone 10: Minimum = 20
- Zone 11: Minimum = 30
- Zone 12: Minimum = 20
- Zone 13: Minimum = 30
- Zone 14: Minimum = 30
- Zone 15: Minimum = 20
- Zone 16: Minimum = 30
Learning parameters optimization for the RBF network
Average quadratic error variation on the test base and for the 16 blocks, methods comparison
Average quadratic error variation on the learning base and for the 16 blocks, methods comparison
Number of pertinent parameters of the neuronal structure for the 16 blocks, methods comparison.
Global output of the system for the noise detection

- Once learning made, images are classified (in network output) following:

  If $\text{abs}(y - 0) < \text{abs}(y - 1)$
  
  $\begin{align*}
  y &= 0 \\
  \text{else} \\
  y &= 1
  \end{align*}$

$y$ : actual output of the network.
RVM, SNN : algorithms (2)

DEMO

function test
The third generation: SNN vs. SVM

1. The successful utilization of SNNs stems from their biological approach to processing information which allows the use of an enormous computing power with small number of neurons which is not the case of SVM.

2. SNN takes some advantages over SVM in particular in the pattern learning approach to detect the uncertainties during prediction which allows to treat a huge number of images with small memory spaces.

3. In SVM, the number of supports vectors will increase with the size of the learning set, the method SNN uses a small number of parameters independently of the size of the learning set.

4. The approach SVM requires to use kernels (radial basis functions) which verify the Mercer theorem, which is not the case in the approach SNN.
Spike neural Network Architecture
Element of Neuronal Dynamics

The state of the neuron is:

\[ u_i(t) = \sum_{t_i^f \in F_i} \eta(t-t_i^f) + \sum_{j \in V_i} w_{ij} \left( \sum_{t_j^f} K(t-t_j^f) \right) + V_{rest} \]

The kernel \( K \) is defined by:

\[ K(t-t_j^f) = V_0 \left( \exp\left( \frac{-(t-t_j^f)}{\tau_m} \right) - \exp\left( \frac{-(t-t_j^f)}{\tau_s} \right) \right) H(t-t_j^f) \]

The kernel \( \eta \) is defined by:

\[ \eta(t-t_i^f) = -\eta_0 \exp\left( \frac{-(t-t_i^f)}{\tau} \right) H(t-t_i^f) \]
Encoding Neurons

Since noise only affects ’L’ component of LAB images, We apply to ’L’ four different de-noising algorithms:

a) Linear filtering with averaging and Gaussian filters to model high frequency noise.

b) Median filtering addresses the salt and pepper noise.

c) Wiener filtering can tailor noise removal to the local pixel variance.

d) The obtained image is denoised also via Wavelet analysis.

e) It has been observed that the wavelet coefficients in the LH, HL and HH sub-bands do not follow a Gaussian distribution.

f) We find that the mean and the standard deviation are important features because their values change until they reach stable thresholds.
Encoding Neurons

- The difference between the features of the sub-image and a quick ray traced sub-image of the scene are provided as inputs to the learning model.

![Graphs and equations related to encoding neurons and standard deviation variation per block.](image)
Spike Neural Network

Centers of Gaussian receptive fields:

Radius of Gaussian receptive fields:

The learning rule is defined:

The parameters of the SNN are defined by:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$\tau_m$</th>
<th>$\tau_s$</th>
<th>$\tau$</th>
<th>$\theta$</th>
<th>$V_0$</th>
<th>$V_{rest}$</th>
<th>$\eta_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>15ms</td>
<td>3.75ms</td>
<td>1ms</td>
<td>1mv</td>
<td>1mv</td>
<td>0mv</td>
<td>1mv</td>
</tr>
</tbody>
</table>
Spike Neural Network

Centers of Gaussian receptive fields:

\[ c = I_{\text{min}}^n + \frac{(2i-3)(I_{\text{max}}^n - I_{\text{min}}^n)}{2(m-2)} \]

Radius of Gaussian receptive fields:

\[ \sigma = \frac{1}{\beta} \frac{(I_{\text{max}}^n - I_{\text{min}}^n)}{m-2} \]

The learning rule is defined:

\[ \Delta w_{ij} = \begin{cases} 
\lambda \sum_{t_f} < t_{\text{max}} K(t_{\text{max}} - t_f) & \text{if } P^+ \\
-\lambda \sum_{t_f} < t_{\text{max}} K(t_{\text{max}} - t_f) & \text{if } P^- \\
0 & \text{otherwise} 
\end{cases} \]

- The parameters of the SNN are defined by:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>( \tau_m )</th>
<th>( \tau_s )</th>
<th>( \tau )</th>
<th>( \theta )</th>
<th>( V_0 )</th>
<th>( V_{\text{rest}} )</th>
<th>( \eta_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>15ms</td>
<td>3.75ms</td>
<td>1ms</td>
<td>1mv</td>
<td>1mv</td>
<td>0mv</td>
<td>1mv</td>
</tr>
</tbody>
</table>
Experimental Results

- For SVM, each of the learning set and the testing set contains 1616 images which are images obtained with different noise levels and reference images.

- In case of SVM, the parameter C is optimized using a Cross-Validation technique.

<table>
<thead>
<tr>
<th>C (ASVM)</th>
<th>Mean Number of SVs</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>441.21</td>
<td>90.22</td>
</tr>
<tr>
<td>2</td>
<td>411.79</td>
<td>89.85</td>
</tr>
<tr>
<td>4</td>
<td>389.77</td>
<td>91.09</td>
</tr>
<tr>
<td>8</td>
<td>366.62</td>
<td>91.89</td>
</tr>
<tr>
<td>16</td>
<td>316.08</td>
<td>92.38</td>
</tr>
<tr>
<td>32</td>
<td>287.30</td>
<td>92.26</td>
</tr>
<tr>
<td>64</td>
<td>258.25</td>
<td>93.25</td>
</tr>
<tr>
<td>128</td>
<td>225.22</td>
<td>93.00</td>
</tr>
</tbody>
</table>

Mean number of SVs and precision for different values of C
Experimental Results

- For SNN, the input of the network is a vector of 182 spikes. The initial weights are selected randomly in the interval $[0, 1]$ and the learning rate is selected equal to 0.003. We find that the weights change drastically for a higher value of learning rate and slowly for a smaller value.
Experimental Results

a) Variation of the average quadratic errors for each sub-image on the learning scene "Bar".

b) Variation of the average quadratic errors for each sub-image on the testing scene "Class".
a) Threshold variation for each sub-image on the learning scene "Bar".
b) Threshold variation for each sub-image on the testing scene "Class".
# Experimental Results

<table>
<thead>
<tr>
<th>Sub-images for Cube</th>
<th>Thresholds</th>
<th>Sub-images for Draper</th>
<th>Thresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2031</td>
<td>2022</td>
<td>2576</td>
</tr>
<tr>
<td></td>
<td>1733</td>
<td>2497</td>
<td>3093</td>
</tr>
<tr>
<td></td>
<td>1653</td>
<td>2660</td>
<td>2967</td>
</tr>
<tr>
<td></td>
<td>2675</td>
<td>2605</td>
<td>3344</td>
</tr>
<tr>
<td></td>
<td>2308</td>
<td>2683</td>
<td>2546</td>
</tr>
<tr>
<td></td>
<td>3413</td>
<td>2180</td>
<td>3072</td>
</tr>
<tr>
<td></td>
<td>1964</td>
<td>2367</td>
<td>3715</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3904</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3922</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3703</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4560</td>
</tr>
</tbody>
</table>

The thresholds obtained by the HVS for the testing scene "Cube“ and “Draper”
Experimental Results

a) Threshold variation for each sub-image on the testing scene "Cube".
b) Threshold variation for each sub-image on the testing scene "Draper".
SNN vs SVM: partial conclusion

1) We introduce the application of SNN to take into account the noise present in global illumination applications.

2) The introduced technique uses fewer parameters than the SVM while offering a good prediction on the testing base.

3) The biggest advantage of our application that this model mimics the human visual system by using a small number of parameters with pattern learning that makes it possible to treat a huge number of images.

4) The generalization of our approach requires establishing a more complete framework with a huge number of images which are difficult to obtain because of the time required for modeling and scene rendering.
RVM, SNN : algorithms (1)
RVM, SNN : algorithms (2)

DEMO

function test
Other possible method: SVM active method (1)

- Differences between learning methods: structures are simplified but error?
- Is it possible to incorporate a-priori knowledge?
Other possible method: SVM active method (2)

- Incorporating diversity in active learning with SVM using a pool-based active learning and SSL
- A set of labeled and unlabeled observations is available
- At each training iteration we use the acquired labels to improve the classifier

- « Semi-supervised and active learning with the probabilistic RBF classifier », C. Constantinopoulos, Neurocomputing 71 (2008)
Semi-supervised and active learning
Other possible method: SVM active method (3)
Other possible method: SVM active method (4)
Partial conclusion

• SNNs are certainly well-adapted to image processing (due to saliency extraction)

• Learning algorithm ?

• Distributed system ?

• Link with HVS ?
Conclusion and future works

1) SVM method is a reliable method for all kind of application (despite of the learning method). Active learning gives very good results!

2) RVM method avoids overfitting (classical neural networks) and takes into account uncertainty (sparse structure: useless parameters disappear).

3) Learning time is about the same for the best structures....
Conclusion and future works

4) Future: deep learning?
Training networks with many hidden layers
Seems computationnally difficult

5) Recurrent NNs to capture the meaningful signal in sequential data and deliver computer-generated descriptions that are remarkably accurate! (J. Donahue, Berkeley)

6) NNs research relies more on experimentation
References


Thanks for your attention!