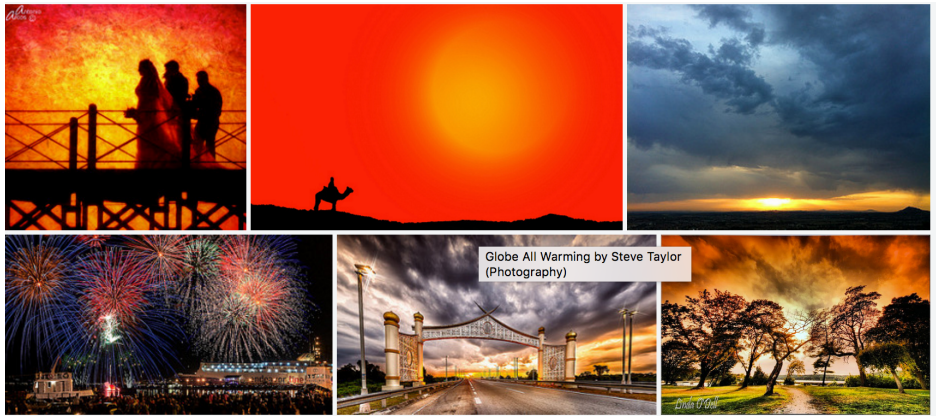
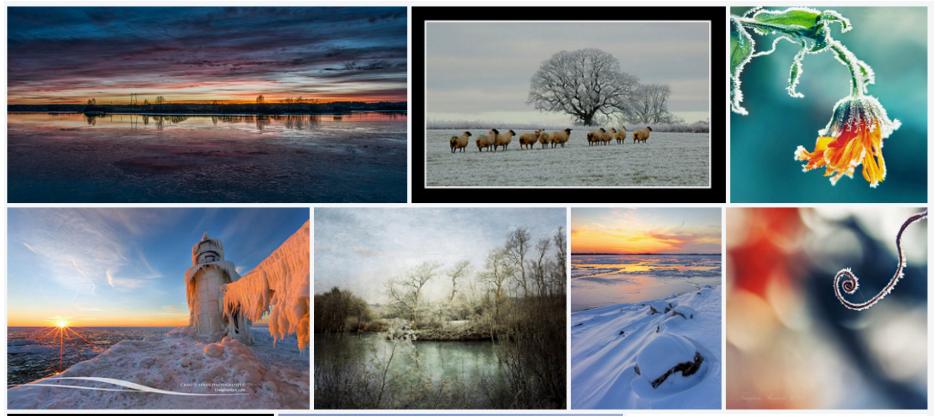
We'll build an application that can "feel" the impression of pictures.

In order to collect the data, we first collect picture data by searching several pair of antonyms; burning <-> freezing, or metropolitan <-> country side.

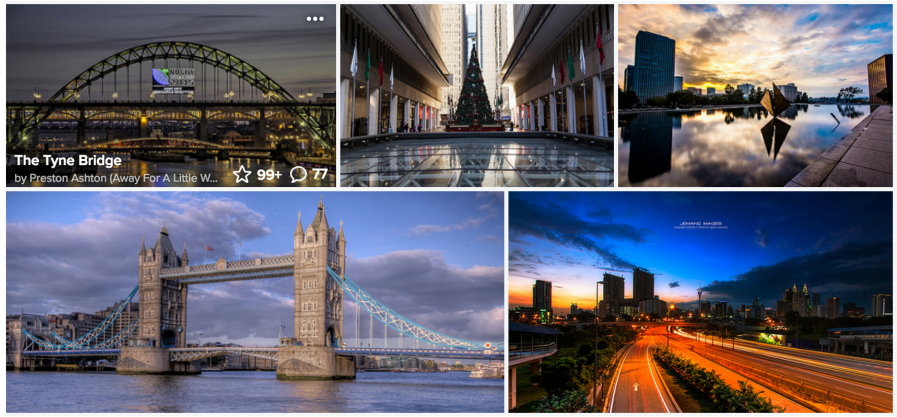
The result of the searches are as follows.



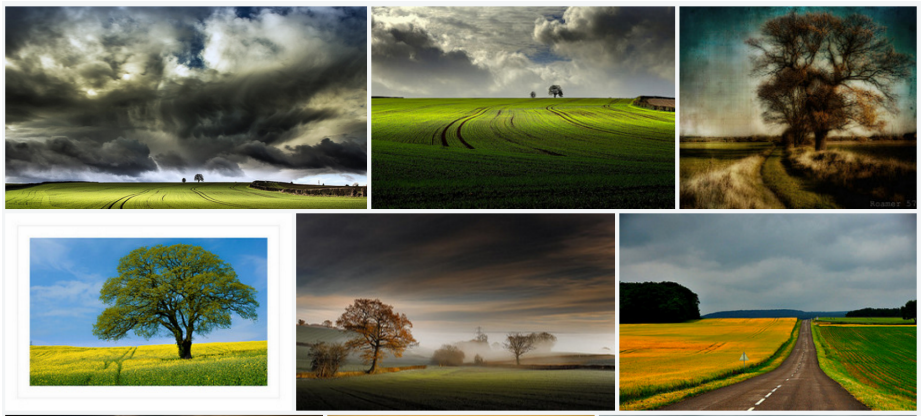
burning



freezing



metropolitan



countryside

The architecture of the system

(1) color histogram

(2) perceptron

metropolitan or countryside

(1) color histogram

To build the color histogram, because the color space is almost continuous space, we'll employ probability density function to smoothen the histogram.

Q1) given that $\mathbf{c}_{(x,y)} = \begin{pmatrix} r_{(x,y)} \\ g_{(x,y)} \\ b_{(x,y)} \end{pmatrix}$

is a color vector of the pixel where $0 \leq x \leq W, 0 \leq y \leq H$

Show the histogram model by using Gaussian-kernel probability density function

using bandwidth from Nonparametric Density Estimation multidim

continued; (1) color histogram

Using the given probability density function at Q1 consumes too much of computer resources,
We'll apply k-means clustering to approximate the function.

Q2) given that k is a number of clusters,

Dividing to bins is easier to implement

$n_i (0 \leq i < k)$ is a number of pixels belonging to i-th cluster,

$\mu_i = \begin{pmatrix} r_i \\ g_i \\ b_i \end{pmatrix} (0 \leq i < k)$ is a mean of points into RGB space,

Show the approximated function of Q1 by using clusters above.

(2) perceptron

The general perceptron model can be represented as:

$$y = f\left(\boldsymbol{w} \cdot \boldsymbol{x}\right)$$

where $\boldsymbol{x}$ is an input and

$$f(a) = \operatorname{sgn}(a) = \begin{cases} 1 & : a > 0 \\ 0 & : a = 0 \\ -1 & : a < 0 \end{cases}$$

however, in our system, the input is not a vector but given as an approximated probability function $p_j(\boldsymbol{s})$ where j represents a picture

Therefore we will extend the concept of inner product of $\boldsymbol{w} \cdot \boldsymbol{x}$ as below

$$g_j = \int_{\mathcal{R}^3} w(\boldsymbol{s}) p_j(\boldsymbol{s}) d\boldsymbol{s}$$

where we rewrite the weight vector as a function as follows

$$w(\boldsymbol{s}) = \sum_{i=0}^{d} \alpha_i \exp\left(-\frac{||\boldsymbol{d}_i - \boldsymbol{s}||^2}{2\sigma^2}\right)$$

where α_i and $\boldsymbol{d}_i$ is initialised randomly then later adjusted in the learning process

Q3) calculate g_j

continued: (2) perceptron — learning part

In order to learn from noisy data, we extend the $f(a)$ into logistics sigmoid function

$$f(a) = \frac{1}{1 + e^{-k(a-a0)}}$$

calculate f() for each images, may choose k = 1, a0 = 0

and learn α_i and $\boldsymbol{d}_i$ by Steepest descent method

$$\Delta\alpha_i = -\eta \frac{f(g_i)}{\partial\alpha_i}$$

use matrix for better run time: log()

$$\Delta\boldsymbol{d}_i = -\eta \frac{f(g_i)}{\partial\boldsymbol{d}_i}$$

Q4) calculate those partial differentiation

Implementation

Assume that the metropolitan <—> countryside is an opposite concept,
just give “1” for metropolitan pictures, “0” for countryside pictures for the output y

Q5) Implement the web system.
  a. Users can upload the picture from the web browser
  b. then the result page will categorise which is the picture in; metropolitan or countryside.

Q6) Improve the accuracy