Image Analogies

Jari Huttunen
48120P
Image Analogies

Jari Huttunen
Helsinki University of Technology

jari.huttunen@hut.fi

Abstract

Image analogies is a framework for processing images by example. Image analogies uses training data \( A \) and \( A' \) in order to learn a filter that can be applied to an unfiltered target image \( B \) to produce an analogous image \( B' \). This means that \( B' \) relates to \( B \) the same way \( A' \) relates to \( A \). The framework is general and can be applied to many different applications. Typical applications for image analogies are texture synthesis, texture transfer, super-resolution, artistic filters, image colorization and texture-by-numbers.

1 INTRODUCTION

Image analogies (Hertzmann et al., 2001) is a framework that uses machine learning and various methods in order to learn and apply filters to an image. The word analogy refers to an attempt to compare two pairs that have the same relationship. The analogy between training data pairs can be used to create new complex image filters. This way we can provide very natural image transformation where the system can learn image filters through example images. In practice image analogies involves two stages. In design phase a designer creates a filter from training images \( A \) and \( A' \). The designer also sets the parameters that control how various type of features in the images are weighted in the image analogy. The filter is then stored to a filter library. Finally in the application phase the end user, probably without further knowledge of image processing, can apply the filter to some image.

2 BACKGROUND

Image analogies is based on various algorithms from different areas. It combines techniques from machine learning, rendering and texture synthesis. Texture synthesis is an area that has been widely researched in the last few years. One of the seminal papers about the texture synthesis is by Heeger and Bergen (Heeger & Bergen, 1995). Their work was very much influenced by using statistics (global histograms of image
features) in texture synthesis. Ashikhmin (Ashikhmin, 2001) introduced just before development of image analogies a texture synthesis method that works very fast and produces good results. In his work correlation between neighboring pixels was used to speed-up the synthesis. The image analogies work is mainly based on this algorithm combined with Wei and Levoy’s work (Wei & Levoy, 2000). Image analogies is closely related to research on scene learning by using Markov Random Fields (MRFs) (Freeman et al., 2000). However, image analogies does not need an iterative algorithm as MRF requires. Image analogies can also provide many new applications that MRFs cannot support like learning artistic filters, where a drawing and painting styles are synthesized based on an example image. The image analogies framework is also related to the previous work dealing e.g. with Video Textures (Schödl et al., 2000), Video Rewrite (Bregler et al., 1997), Voice Puppetry (Brand, 1999) and Style Machines (Brand & Hertzmann, 2000).

3 IMAGE ANALOGIES FRAMEWORK

This section describes the image analogies framework, its current problems, and improvements that could be done in the future.

3.1 Overview

The image analogies framework includes two phases, a design phase and an application phase in order to produce the desired output image. In design phase a pair of images is used as a training data for the algorithm where the other one is the original image and the other is a filtered version of that one. This way the the framework produces a new filter that can be used in the application phase to produce an analogous filtered image. The resulting filtering effect can be modified by choosing different source image pairs as input and by modifying the algorithm parameters.

3.2 The algorithm

The image analogies algorithm takes as an input three images, namely the unfiltered source image $A$, the filtered source image $A'$, and the unfiltered target image $B$. As an output the algorithm produces the filtered target image $B'$. The algorithm assumes that the colors in $A$ and $A'$ correspond to each other in the same pixel position $p$. The position of $p$ in source picture is stored in its own data structure $s()$. It will be copied to pixel $q$ in target picture Therefore, we will have the property $s(q) = p$. The images need to contain the information of RGB color, luminance and various filter responses. This information combined we will have the feature vector for each pixel. The feature vector can be noted as $A(p)$ for original source image or $A'(p)$ for filtered source image. For the target image the feature vector is respectively $B(p)$ for the unfiltered
and \( B'(p) \) for the filtered image. The feature selection is described in more detail in section 3.3.

The algorithm initializes by creating multiscale (Gaussian pyramid) representations of \( A, A' \) and \( B \). Also feature vectors and some additional indices for speeding up the algorithm are created. The synthesis of \( B' \) then proceeds from coarsest resolution to finest, computing the multiscale representations of \( B' \) for each level. At every level the “best match” is examined by comparing the statistics for every pixel \( p \) in source pair to statistics in each pixel \( q \) in the target pair. The pixel that matched best is stored in \( s_\ell(q) \) and the feature vector \( B'_\ell(q) \) is set to the feature vector \( A'_\ell(p) \). The best match is found using two different search approaches, an approximate search and a coherence search. The approximate search uses feature vectors of \( p, q \) and their neighborhoods to find the pixel that matches the best. Two algorithms have been tested by Herzmann et. al (Hertzmann et al., 2001) to solve the approximate search problem: Approximate-nearest-neighbor search (ANN) (Arya et al., 1998) and tree-structured vector quantization (TSVQ) (Gersho & Gray, 1992). ANN produces in general more accurate results for the same computational time, although it is very memory intensive. The image analogies software, described in section 5, uses ANN to find the best approximate match. The coherence search is based on Ashikhmin’s approach (Ashikhmin, 2001) and attempts to preserve coherence with the neighboring synthesized pixels. The approximate-search distance is rescaled using a coherence parameter \( \kappa \). The larger the value of \( \kappa \) is, the more coherence is favored. The coherence parameter \( \kappa \) is attenuated by a factor \( 2^{\ell-L} \) in order to keep the coherence term consistent at different scales.

### 3.3 Feature selection and representation

Selecting and representing features is the main open problem in machine learning. The use of RGB channels is usually the main choice to try to solve the problem. However, for some filters the source image pairs do not include enough data to match the target pair when using only RGB color. Therefore, the image analogies framework uses also other alternative solutions for feature selection. One alternative solution is to compute and store the luminance at each pixel and use it instead of RGB in the distance metric. This approach is motivated by the fact that we are more sensitive for changes in the luminance channel than in color difference channels. The luminance can be computed many ways; image analogies uses \( Y \) channel from the YIQ color space and I and Q channels for color difference components. After the processing is done in luminance space, the color information is recovered by copying the I and Q channels of the input \( B \) image to the target \( B' \) image. This approach can help also in speeding up the matching and synthesis procedure. However, in some cases converting to luminance can still give poor results. This can happen e.g. when a light source image \( A \) is used to process dark \( B \). Image analogies uses linear mapping that matches the means and variances of the luminance distributions. The luminance remapping can be done as

\[
Y(p) \leftarrow \frac{\sigma_A}{\sigma_B}(Y(p) - \mu_A) + \mu_B
\]
where $Y(p)$ is the luminance of a pixel in image $A$, $\mu_A$ and $\mu_B$ are the mean luminances, and $\sigma_A$ and $\sigma_B$ are the standard deviations of luminance.

### 3.4 Problems and limitations

The biggest problem in image analogies framework is probably the slowness of the algorithm. The performance of the algorithm is logarithmic with respect to the size (in pixels) of the training pair and linear with respect to the size of the target. Even though the algorithm uses several techniques to speed up the process it is still quite slow and far from real time. Even for simple texture synthesis it takes tens of seconds to accomplish the task. To do artistic rendering it can take already some hours on a PC with 1 GHz processor and require hundreds of megabytes of RAM. The memory intensiveness is largely due to the use of ANN in approximate search. TSVQ would be a more memory friendly solution, but it would require even more processing power. The inefficiency of the implementation is typical for a method in prototyping phase. The image analogies framework has never been really tuned since the focus has been in testing different algorithms. With optimizing the algorithm the procedure can be speeded up. Also the search and regression techniques used in the algorithm are quite inefficient and would need some improvement.

As mentioned in section 3.3 the feature selection is problematic since each approach fails in some situation. The image analogies algorithm uses the best properties of $L_2$ metric and Asikhmin’s algorithm, but still it produces some artifacts. Therefore, there is a need to find better algorithms for feature selection and representation.

The pointwise correspondence between the training images is also a problem in image analogies technique. It would be desirable if the algorithm could automatically learn the correspondence between pixels and maybe even apply automatically the learned distortion during synthesis.

### 4 APPLICATIONS

Image analogies can be used for many different applications. Here we go through some applications where image analogies has been applied so far. All filtered target images shown in this paper are created by using the Image Analogies software (copyright © Aaron Hertzmann and Chuck Jacobs.)

#### 4.1 Image filters

A simple application for image analogies are image filters, e.g., a “blur” or a “sharpen” effect. The image analogies technique gives quite good results in image filtering but is rather inefficient. Applying the filter directly as is usually done in a graphic software (e.g. in Adobe Photoshop) is much more efficient.
4.2 Texture synthesis

In texture synthesis a texture that "looks like" some example texture is created, without being a copy of the example. Efros and Freeman (Efros & Freeman, 2001) have introduced an efficient texture transfer method called image quilting. Image quilting is a fast and simple process where a new image is synthesized by stitching together small patches of existing image. The image quilting method can be extended also to texture transfer described in section 4.3. In image analogies framework texture synthesis can be thought as a trivial case where the unfiltered images A and B are zero-dimensional or constant. Image analogies combines the advantages of the weighted $L_2$ norm and Ashikmin's search algorithm (Ashikhmin, 2001) excluding the speed of the algorithm. An example of texture synthesis is shown in figure 1.

![Image of texture synthesis example](image1.png)

Figure 1: An example of texture synthesis. The input image is filtered using $\kappa=5$. (Texture copyright © 2004 Jari Huttunen)

4.3 Texture transfer

Texture transfer means basically rendering an object with a texture taken from a different object. It means that the image B is filtered so that it has the texture of a given example image A'. A variable $w$ noted as weight can be used to describe the similarity between the source images ($A, B$) and the target images ($A', B'$). Increasing $w$ causes the source image to be reproduced more faithfully, while decreasing $w$ ties the image more closely to the texture. If $w$ equals 0, the texture transfer falls back to ordinary texture synthesis. In figure 2 a rice texture is transferred to a photograph of a statue using the source weight $w = 0.2$. 

![Image of texture transfer example](image2.png)
Figure 2: An example of texture transfer. Source texture is used both as A and A’.

(Copyright © 2004 Jari Huttunen)

4.4 Artistic filters

Artistic filters are designed to add some abstraction to the image. Sometimes it is used in order to approximate a certain real-life effect (such as painting and brush techniques, murals, and so forth), and sometimes to create something new and unique. When creating, e.g., a painting effect we can use as a training pair a photograph of an original picture A from where the painting was made and the painting A’ made by the artist. If there is no source photograph available, a substitute has to be created by hand. In figure 3 an example is shown where we have used the ‘Smart Blur’ filter from Adobe Photoshop to create A from A’ since the original picture was not available. The use of photograph and a painting as a training pair is problematic. The training pairs A and A’ need to be very carefully in pointwise correspondence in order to get successful results. Therefore, in practice some alignment processing to the images needs to be done before applying artistic filtering.

4.5 Texture-by-numbers

In texture-by-numbers the texture components of the original source image are labeled by hand and the resulting statistics are applied to a new labeling image B. The advantage of using texture-by-numbers is that it can give better results than texture transfer in some cases. Texture-by-numbers has been used for combining aerial maps and land surveys. An example is shown in figure 4 where filtered terrain image B is achieved by using A, A’ and B as training data.
Figure 3: Artistic filters. The style from the painting by Marjatta Huttunen has been copied to the target photograph B using coherence parameter $\kappa=5$. (Scenery photograph copyright © 2004 Suvi Räisänen)

### 4.6 Super-resolution

Super-resolution is an intelligent method for expanding the resolution of an image. It is used to create high-resolution detail from low-resolution images. The algorithm takes a low resolution image $A$ and a high resolution image $A'$ as an input and creates a new high resolution image $B'$ from low resolution image $B$. Usually only one image pair is used as a training material for the algorithm but image analogies framework supports also multiple training pairs.

### 4.7 Image colorization

The image analogies algorithm can also be used to apply color information to a grayscale image. An unfiltered grayscale source $A$ and colorized source image $A'$ are used to produce a filtered color image $B'$ from gray scale image $B$. See figure 5 for an example. Similar technique has been developed by Ashikmin et. al (Ashikhmin et al., 2002)
where the entire color mood is being transferred from the source to the target image by matching luminance and texture information between the images. In Ashikmin’s approach only the chromaticity information is transferred and the original luminance values of the target image are retained.

5 Image analogies software

The software called "lf" ("learn filter"), used to produce the output images in this paper and in Herzmann’s image analogies paper (Hertzmann et al., 2001), can be freely downloaded from the image analogies project web page at http://mrl.nyu.edu/projects/image-analogies/lf/. The software is research code and
therefore badly documented and there are no guarantees that it works. The lf includes a
makefile for Unix and Windows environment. Executable binaries are provided in the
distribution for Linux and Windows operating systems. The lf uses many free libraries
like OpenGL, GLUT, GLUI, libPNG, ANN and LAPACK. Most of these libraries are
common in Linux distributions and most or all libraries needed for Windows are in-
cluded to the lf distribution.

The Windows version of the lf software was used to produce the output images in
this paper. The software includes many options to be given as an argument. Unfortu-
nately none of these options have been documented so far. Luckily, the lf distribution
includes some examples that can be used as a template to create own images. The
examples in the distribution are configuration text files that can be loaded to the lf
software. The most important parameter in configuration files is the “coherenceEps”
parameter (defined as $\kappa$ in section 3.2). The input images must be in PNG, PPM or
BMP format. If the input images are large and they include color information, the lf
is a very memory and CPU time intensive application. The images used in this paper
were size of 480*360 pixels with 16 million colors. The images were processed with
If software using a PC with 1466 MHz processor and 256 mega bytes of DDR RAM. The amount of RAM was not enough because the average memory consumption was usually over 300 mega bytes. Hence, the processing of the images was quite slow as virtual memory had to be used to extend the RAM.

A somewhat similar software as image analogies if is the Paul Harrison’s Resynthesizer. It is a GIMP plug-in for texture synthesis. Resynthesizer works by piecing together a new image from pieces of the input texture, one pixel at a time (Harrison, 2001).

6 CONCLUSIONS
Image analogies is a versatile framework that can be used for wide variety of image texturing problems. For basic image filtering image analogies is rather inefficient and better results will be obtained by applying the filter directly. Texture transfer and texture synthesis are an example of applications that suit well for the image analogies framework. The generality of the framework makes it good for future research in order to extend it to solve more problems, even outside the image processing area.

REFERENCES


