Experiments in Example-based Image Filter Retrieval

Pedro Torres*, Simon Colton*, and Stefan Rüger+

*Department of Computing, Imperial College, London, UK. ptorres,sgc@doc.ic.ac.uk
+Knowledge Media Institute, The Open University, UK. s.rueger@open.ac.uk

Abstract. We present a novel image filter generation method and an image filter retrieval algorithm and analyse their properties. Based on an original image and a filtered version of the original, the retrieval algorithm can find, to a high probability, which filter was applied to the original from a large pre-defined list of filters, without having to apply all filters to the original image, which is usually a time consuming task when the number of filters is large. This is achieved by pre-computing image annotations for a set of filtered images obtained by applying the pre-defined filters to a database of 50 images. Using standard image-based annotation techniques, we show that the filter retrieval can be achieved by taking the closest images to the original from the database and analysing those known images instead. The retrieval algorithm has a set of parameters and we present results of experiments with these values to maximise the probability of retrieving the correct filter.

1 Introduction

Image filtering [4] is the process whereby the pixel information of a digital image is mathematically manipulated to produce an altered version of the image. Image filtering is a very commonplace process in graphic design, and modern software packages such as Adobe Photoshop include hundreds of image filtering techniques ranging from simple edge detection to the production of artistic images simulating watercolour effects, etc. Along with other graphics packages, Photoshop enables browsing through the range of filters and their parameterisations. We address here the question of searching through a database of image filters to find a match given an image and a filtered version of the image, or given only the filtered version of the image. Potentially, this may enable a novice user to more easily produce an image effect, because they can supply an exemplar of the kind of effect they wish the image filter to produce, and experiment with the image filters which are returned from the search, rather than having to browse through sequences of filter applications. Another potential application of the retrieval algorithm described here may be medical imaging [1]. Here, the system could be used to retrieve filters which enhance certain characteristics of an image under analysis, so that a particular aspect of the image becomes more prominent.

To determine good methods for searching for filters, we work with 1000 image filters of our own devising, as described in §2. In §3, we describe the retrieval

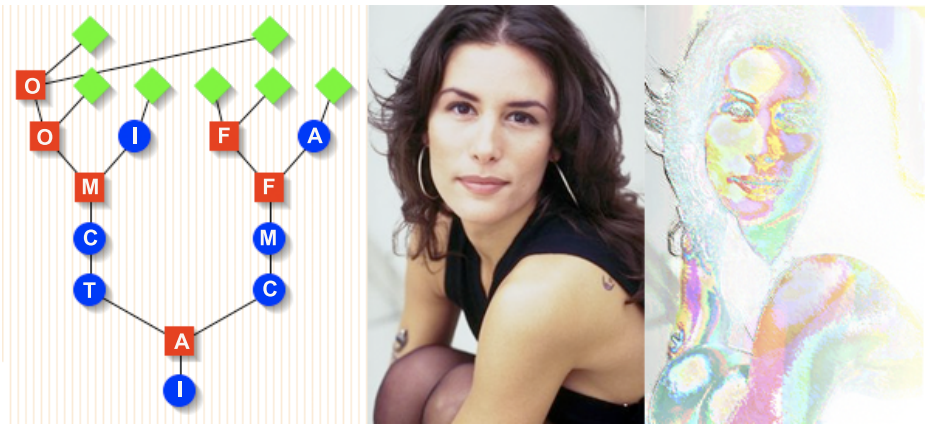

Fig. 1. Example image filter tree consisting of transforms in blue circles: A=Add Colour, C=Convolution, I=Inverse, M=Median and T=Threshold; and compositors in red squares: A=And, F=Fade, M=Min and O=Or. Image inputs are in green diamonds. An example original image and the filtered version are shown.

methods we have experimented with, and in §4, we describe our experiments and the results. We conclude that – at least with the 1000 image filters we have been working with – it is possible to retrieve a filter given a single image produced by it, to a high degree of probability. We further discuss some future directions, including the application of machine learning techniques to this problem.

2 Image Filter Database

Our database consists of 1000 image filters, each of which can be represented as a tree of fundamental (unary) image transform such as inverse, lookup, threshold, colour addition, median, etc., and also (binary) image compositors such as add, and, divide, fade, max, min, multiply, or, subtract, xor, etc., as described in [5] and [7]. An example tree is provided in figure 1. We see that the overall filter uses seven transform steps and six compositor steps, and that the original image is input to the tree seven times. The 1000 filters were produced by randomly generating a tree structure subject to a limit on the number of nodes (usually 30), with each transform and compositor added to the tree with random parameter settings, e.g., the median transform takes two parameters, the first of which determines the way in which the median RGB values of each pixel is calculated, and the second of which determines the extent of the neighbourhood that is taken into account when the median is calculated. Each time an image filter generated in this random way produced an interesting visual effect on one or more original images, it was added to the database. The time taken to apply a filter is roughly proportional to the size of the input image and the size of the tree. Over the entire database of filters, the average number of nodes in a tree is

13.62, and the average time[1] to apply a filter to an image of dimension 256 by 384 pixels (the size used for the experiments reported here) is 410 milliseconds. Hence it would take a little under 7 minutes to apply the entire set of filters to an image of this size. In practice, images tend to be larger than this, and hence the application of all the filters to an image for search purposes is infeasible.

3 Retrieval Techniques

3.1 Definitions

An *image* is a function $I : \{0, \ldots, N\} \times \{0, \ldots, M\} \rightarrow \{0, \ldots, 255\}^3$, for some N and M. The set of all images will be denoted by $\mathcal{I}$. A *filter* is a function $F : \mathcal{I} \rightarrow \mathcal{I}$. The set of all filters will be denoted by $\mathcal{F}$. The symbol $F(I)$ will then stand for the *application of filter F to image I*.

An *annotation*[2] *of dimension d* is a function $A : \mathcal{I} \rightarrow \mathbb{R}^d$. The set of all annotations of dimension d is denoted by $\mathcal{A}_d$ and *the set of all annotations* is defined as $\mathcal{A} = \cup_{d=1}^{\infty} \mathcal{A}_d$. Writing $P_z = \{(w_1, \ldots, w_z) \in \mathbb{R}^z | \forall i. 0 \leq w_i \leq 1 \wedge \sum_{i=1}^{z} w_i = 1\}$, we define a *set of weighted annotations* to be a pair $(\mathcal{A}_0, w)$ such that $\mathcal{A}_0 \in \mathcal{A}$ and $w \in P_{|\mathcal{A}_0|}$. The set of all weighted annotations is denoted by $\mathcal{W}$.

Let I be an image, $(\mathcal{A}_0, w)$ a set of weighted annotations and $A \in \mathcal{A}_0$ an annotation of dimension d. The $\mathbb{R}^d$ vector $A(I)$ is called the *annotation of I via A* or an *A-annotated image*. We can introduce metrics in the space of A-annotations using any existing metric for $\mathbb{R}^d$. For example, we may define the distance between two images, I and J, as the weighted Euclidean distance between pairs of real vectors $A_k(I)$ and $A_k(J)$ for each $A_k \in \mathcal{A}_0$:

$$D_{(\mathcal{A}_0, w)}(I, J) = \sum_{k=1}^{|\mathcal{A}_0|} w_k \cdot d(A_k(I), A_k(J)) \tag{1}$$

where d is the usual Euclidean distance. We will use the symbol 2^X, where X is a set, to denote the power set of X.

3.2 Problem Formulation

The problem we address here is as follows: given (i) an image $I \in \mathcal{I}$, (ii) a set of filters $\mathcal{F}_0$ and (iii) a filtered version of I, $I' = F(I)$, for some $F \in \mathcal{F}_0$, determine the filter F. A straightforward naïve algorithm to solve this problem is shown below. The NAÏVE algorithm simply applies all filters to the user image and checks which filter achieves the same end image. The problem with this naïve algorithm is that we are generally interested in the case where $\mathcal{F}_0$ is a large set

[1] On a MacOs X machine running at 2.6Ghz.

[2] An annotation is a vector of what are usually called *features of an image*, which in turn are real values computed from that image.

FILTER($\mathcal{I}_D : 2^{\mathcal{I}}, \mathcal{F}_D : 2^{\mathcal{F}}) : 2^{\mathcal{I}}$

1. $\mathcal{I}_F \leftarrow \emptyset$
2. **for each** $I \in \mathcal{I}_D$
    1. **for each** $F \in \mathcal{F}_D$
        1. $\mathcal{I}_F \leftarrow \mathcal{I}_F \cup \{F(I)\}$
3. **return** $\mathcal{I}_F$

NAÏVE($\mathcal{F}_D : 2^{\mathcal{F}}, I_U : \mathcal{I}, I'_U : \mathcal{I}) : \mathcal{F}_D$

1. $\mathcal{I}_F \leftarrow$ FILTER($\{I_U\}, \mathcal{F}_D$)
2. **for each** $I' \in v_F$
    1. **if** $I = I'_U$
        1. $F \leftarrow$ filter which produced I'
        2. **return** F

and applying thousands of filters to an image is very time consuming. Moreover, we are interested in finding an efficient algorithm for the above problem for the case where the filter retrieval operation is to be performed repeatedly. So, we can concede some initial offline computation time if we can have fast retrieval thereafter.

3.3 Filter Retrieval Algorithm

We will now present a general algorithm to perform such filter retrieval, which is described below in pseudo-code. We start with a database of images, $\mathcal{I}_D$, and filters, $\mathcal{F}_D$, a set of weighted annotations $(\mathcal{A}_0, w)$ and three integers which parameterise the algorithm. We start by choosing a random sample of size O of the image database $\mathcal{I}_D$ and choose the N images of that sample which are closest to the user image, I_U. We denote this set of closest images by C. Note that the distance between two images I and J is measured here as the weighted Euclidean distance between their annotations obtained via $\mathcal{A}_0$, as described by Equation (1). We then retrieve the filters which, applied to the images in C, yield images closest to the user-given filtered image, I'_U. This set of filters $\mathcal{F}_R$ is then returned. Given an integer K, a set X and a function $f : X \to \mathbb{R}$. The set $\text{argmin}^K_{x \in X}\{f(x)\}$ represents the K values of x which yield minimum values for $f(x)$. If multiple values of x yield the same minimum value for f, we take the resulting set to be the union of all those values of x.

RETRIEVE($\mathcal{I}_D : \mathcal{I}, \mathcal{F}_D : \mathcal{F}, (\mathcal{A}_0, w) : \mathcal{W}, (N, O, T) \in \mathbb{N}^3) : \mathcal{F}_D$

1. $S \leftarrow$ RANDOMSAMPLE($O, \mathcal{I}_D$)
2. $C \leftarrow \text{argmin}^N_{I \in S}\{\sum_{k=1}^{|\mathcal{A}_0|} w_k \cdot d(A_k(I), A_k(I_U))\}$
3. $\mathcal{F}_R \leftarrow \text{argmin}^T_{F \in \mathcal{F}_D}\{\sum_{I \in C} \sum_{k=1}^{|\mathcal{A}_0|} w_k \cdot d(A_k(F(I)), A_k(I'_U))\}$
4. **return** $\mathcal{F}_R$

In the algorithm, given an image I, we write $A_k(I)$ to be the kth-annotation of I. Note that, in this algorithm, all images in the database have been previously filtered and both original and filtered images have been annotated beforehand. This means that at retrieval time both $A_k(I)$ and $A_k(F(I))$ are known and no calculations are needed other than Euclidean distances between vectors. The only annotations to compute at retrieval time are $A_k(I_U)$ and $A_k(I'_U)$.

4 Experimental Results

The RETRIEVE algorithm can be assessed by choosing particular parameters and comparing the retrieval accuracy and retrieval time. For $\mathcal{I}_D$, we hand-picked 50 images from the Corel Stock Photo Library illustrating a wide range of everyday images, including portraits and full-body pictures of men and women, cityscapes and countryside scenes, among others. For $\mathcal{F}_D$ we took the set of 1000 filters described in §2. These filters represent a broad sample of the filter space $\mathcal{F}$.

The set of annotations $\mathcal{A}_0$ chosen consists of four main modes of operation of the Annotate software, which has been used as an automated image annotation system using global features in [6]. These were: (i) *gabor* with s=4 and o=6, (ii) linear HSV histograms with $4 \times 4 \times 4$ sampling, (iii) RGB histograms with $4 \times 4 \times 4$ sampling and (iv) *tamura* with 3×3 tiling. The exact Annotate command-line options for these modes were: `-a gabor-4-6`, `-a HSV_lin-4x4x4-G`, `-a RGB-4x4x4-G`, `-a tamuraS-2-A.T3x3`.

Having made these choices, the only remaining parameters of the RETRIEVE algorithm were N, O, T and the set of four weights, w. To study this space of parameters, we implemented the RETRIEVE algorithm and, for different choices of the parameters, we measured the proportion of times that the correct filter was found in the set of filters returned by the algorithm.

We started by studying the space of weights, $[0,1]^4$, for the particular setting $(N, O, T) = (3, 50, 10)$, in which we considered the whole database of images $\mathcal{I}_D$ ($O = 50$) chose the closest 3 images ($N = 3$) and retrieved 10 candidate filters ($T = 10$). Each test for a particular set of parameters (N, O, T, w) takes between several minutes to more than an hour. Hence, we decided to explore the space of weights non-exhaustively by hand. From this exploration, we concluded that a good set of weights was $w = (0.005, 0.465, 0.530, 0)$. It is not guaranteed to be the best choice of weights but – as we see below – with this choice, the retrieval method will find the correct filter in a set of 10 retrieved filters nearly 93% of the time. Hence, these weights gave us a good starting point to explore the remaining three parameters. The weights chosen reflect an almost identical weight of RGB and HSV annotations and a small contribution of the gabor annotation; the inclusion of the tamura annotation did not seem to increase the probability of correct retrieval, but this can only be confirmed by a more exhaustive search of the weight space.

For this particular set of weights, $w = (0.005, 0.465, 0.530, 0)$, we assessed how the remaining parameters affected the probability of correct retrieval, with the results given in figure 2. We carried out three different sets of experiments: (i) we fix $O = 10, 50$ and computed the probability of correct retrieval for varying $N = 1, 2, 3, 5, 10, 25, 50$ and $T = 1, 5, 10, 25, 50$ (each different value of T produces one curve and each choice of O is shown in a different graph); (ii) we fixed $T = 10, 50$ and computed the probability of correct retrieval for varying $N = 1, 2, 3, 5, 10, 25, 50$ and $O = 1, 2, 3, 5, 10, 25, 50$ (each different value of O produces one curve and each choice of T is shown in a different graph); (iii) we fixed $N = 2$ and computed the probability of correct retrieval for varying $O = 2, 3, 5, 10, 25, 50$ and $T = 1, 5, 10, 25, 50$ (each different value of T produces

one curve). In the sixth graph of figure 2, we show the retrieval times which correspond to the setting of experiment (i) with $O = 10$.

From the first set of experiments (fixed O) we see that there is a maximum for the parameter N and that it is generally attained for either $N = 2$ or $N = 3$. This is particularly clear in the $O = 50$ graph. So, we conclude that it is good to have more than one image to compare the user's image to but that using more than 3 images degrades the subsequent comparison of annotations. We can also confirm that increasing T always improves retrieval success. This was expected since returning more filters increases the probability that they contain the correct filter but it shows us that the difference in results from taking $T = 25$ or $T = 50$ is small. Even for $T = 10$, the difference is still acceptable and this choice seems to be a good compromise between probability of correct retrieval and the amount of filters that the user still has to choose from after the retrieval. For $(N, O, T) = (3, 50, 10)$, the probability of the required filter being present in the 10 retrieved filters is 0.928.

From the second set of experiments (fixed T), we see that increasing O always achieves better results, which is expected since greater O means a larger image database to work with and therefore more likelihood of finding an image which is closer to the image provided by the user. Once again, there is a maximum for N around 2 or 3 except for very small O where the initial database is so small that in fact, it is better just to keep the closest image in the database. From the third experiment (fixed N) we see that for $N = 2$, both increasing either T or O always increases retrieval success as before. In the sixth graph in figure 2 we present retrieval times corresponding to the first experiment with $O = 10$. We see that retrieval times lie in the range $10 - 40$ ms, are hardly affected by T, and increase roughly linearly with N. For larger values of O (not shown) retrieval times increase considerably but never raise above 300 ms. We will not therefore emphasise comparison between retrieval times for the different parameters as for practical purposes of user retrieval, the times are small (between $10 - 300$ ms). We did not compute retrieval time for the NAÏVE algorithm as it is clear from the algorithm's definition that retrieval times will be dominated by the filter application times which, when applied 1000 times, will add up to several minutes. In future, we will apply the NAÏVE algorithm to the filters returned by RETRIEVE, which will improve accuracy, but only degrade efficiency slightly.

Finally, we considered the scenario $N = O$ separately. This can be seen as equivalent to the case where the user only provides the RETRIEVE algorithm with the filtered image I'_U but not the original image I_U. If we do not know the original image, we can simply pick one or more random images from the database and compare their filtered versions to the filtered user image. This amounts to changing the second line of the RETRIEVE algorithm to $C \leftarrow S$. The results of this case are shown in the final graph of figure 2 for $N = O = 1, 2, 3, 4, 5, 10, 20, 30, 40, 50$ and $T = 1, 5, 10, 25$. We see that considering more than 5 images in the database most of the times does not add to the retrieval probability and that for $T = 10$, the probability now drops to around 0.727, but if the user chooses from 50 retrieved filters ($T = 50$), the probability is 0.914.

5 Conclusions and Future Work

We have presented an image-filter retrieval algorithm which takes a user-given image and filtered version of the image and retrieves a set of 10 filters (from 1000) which contains the applied filter with a probability of 0.928 in a fraction of a second. The retrieval algorithm has 7 parameters and we have showed how its behaviour is affected by changing these parameters. We find the best choice for 3 parameters and near best choices for the remaining 4. We also show that the method can retrieve the correct filter even if the original (unfiltered) image is not given, albeit with a smaller probability of success.

In future, we plan to study how different distance metrics in step 2 of RETRIEVE affects its performance. Moreover, we will study how the choice of image database may influence the probability of correct retrieval by repeating the same experiments with different sets of images. We also plan to address the more general problem of learning a set of properties for a required filter (or filters) given a set of images and/or filters that the user has chosen through a browsing mechanism. In particular, we will use mathematical theory formation [3] and closed-loop learning [2] to approximate a user's aesthetic preferences for image filtering during a session. These preferences will then be used as a fitness function to search for and/or evolve image filters that may be of interest to the user. In addition, we will investigate any benefit of our approach to medical image filtering, and by replacing the compositors and transforms in our tree structure by Photoshop actions, we plan to show that AI techniques can be used to increase the benefit of graphic design software to designers and artists.

Acknowledgements

We wish to acknowledge the excellent work of Imperial College masters students George Lin and Chi-Tou Yip for their investigations into evolving image filters. We would also like to thank João Magalhães for his extensive input on the usage of the Annotate software. This research is funded by EPSRC grant EP/F067127.

References

1. C Behrenbruch, S Petroudi, S Bond, J Declerck, F Leong, and J Brady. Image filtering techniques for medical image post-processing: an overview. *British Journal of Radiology*, 77(2).
2. C Bryant, S Muggleton, C Page, and M Sternberg. Combining active learning with inductive logic programming to close the loop in machine learning. In *Proceedings of the AISB'99 Symposium on AI and Scientific Creativity*, 1999.
3. S Colton. *Automated Theory Formation in Pure Mathematics*. Springer, 2002.
4. R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2008.
5. Z. Lin. Evolving image filters. Master's thesis, Imperial College London, 2005.
6. A Yavlinsky, E Schofield, and S Rüger. Automated image annotation using global features and robust nonparametric density estimation. In *Proceedings of the International Conference on Image and Video Retrieval*, pages 507–517. Springer, 2005.
7. C. Yip. Evolving image filters. Master's thesis, Imperial College London, 2004.

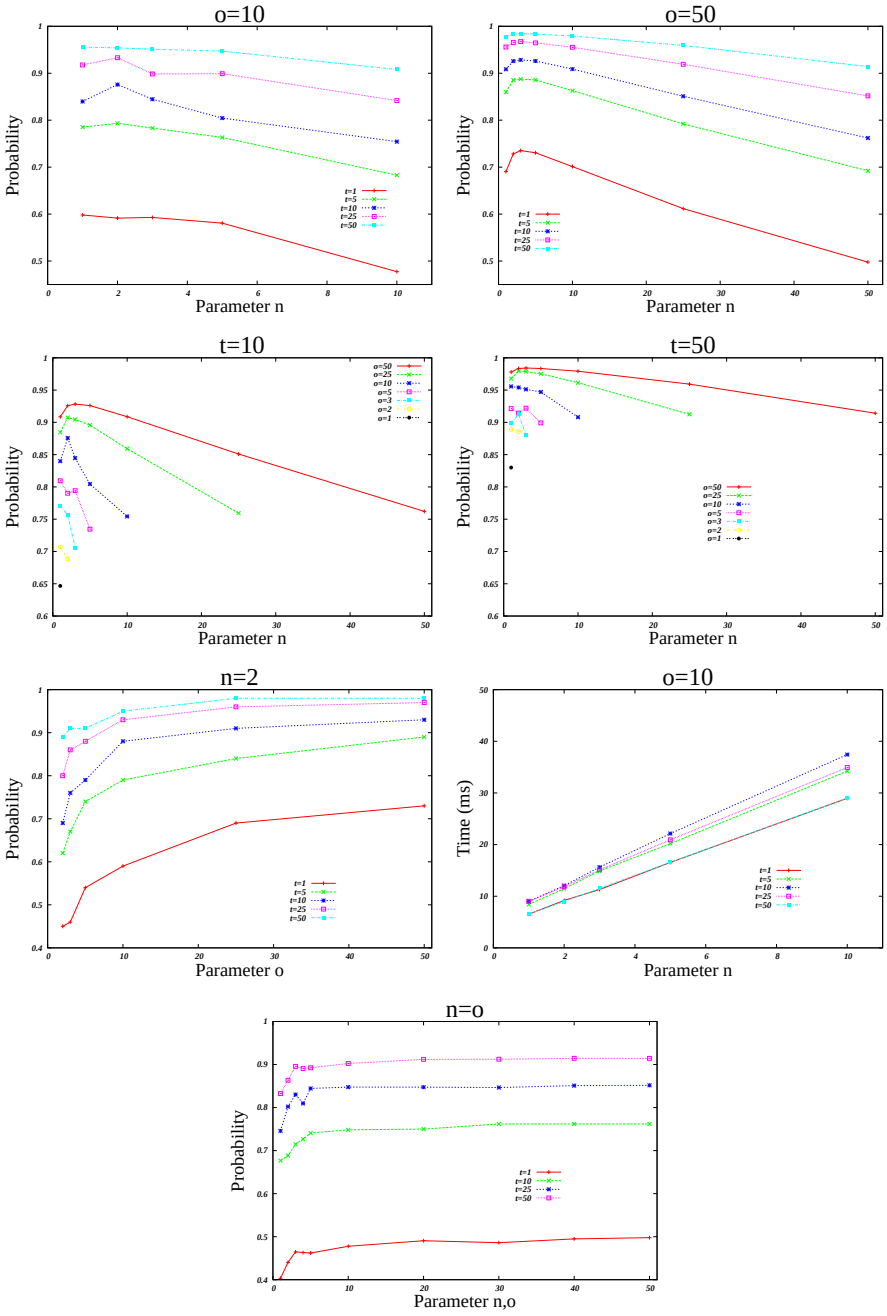

Fig. 2. Probability of correct retrieval for experiments (i), (ii) and (iii) (first five graphs) and retrieval times of $O = 10$ (sixth graph). The weights were set to $w = (0.005, 0.465, 0.530, 0)$. Final graph: the probability of correct retrieval for $N = O$.