Abstract Art by Shape Classification

Yi-Zhe Song, David Pickup, Chuan Li, Paul Rosin, and Peter Hall

Abstract—This paper shows that classifying shapes is a tool useful in Non-Photorealistic rendering from photographs. Our classifier inputs regions from an image segmentation hierarchy and outputs the "best" fitting simple shape such as a circle, square or triangle. Other approaches to NPR have recognised the benefits of segmentation, but none have classified the shape of segments. By doing so, we can create artwork of a more abstract nature, emulating the style of modern artists such as Matisse and other artists who favoured shape simplification in their artwork. The classifier chooses the shape that "best" represents the region. Since the classifier is trained by a user, the 'best shape' has a subjective quality that can over-ride measurements such as minimum error and more importantly captures user preferences. Once trained, the system is fully automatic, although simple user interaction is also possible to allow for differences in individual tastes. A gallery of results shows how this classifier contributes to NPR from images by producing abstract artwork.

1 Introduction

Image-based Non-Photorealistic Rendering (NPR) aims to process photographs into artwork. The problem is difficult for many reasons that are not limited to the variety of implements used to make marks (*e.g.* pencil, brush), the variety of media (*e.g.* oil, watercolor), and the variety of substrates (*e.g.* paper, canvas). In addition, the wide gamut of distinct styles or schools of artwork means that the literature in NPR comprises a collection of techniques each of which are intended to emulate a small subset of styles.

Initial research in image based NPR used filters to emulate brush strokes, a tradition that continues. Filtering is able to emulate styles such as impressionism and other schools that are figurative in nature. However, many schools of art are characterised by changes to the underlying geometry of parts and the relationship between the parts. Examples can be found throughout history (*e.g.* Egyptian art, Mediaeval Art, Cubism) and across the world (*e.g.* Chinese Art, African Art). These styles cannot be emulated by filtering alone.

The contribution of this paper is to provide NPR from images with a method for producing abstract artwork that has hitherto proven impossible. Specifically, we use image regions as graphic primitives; the regions are detected using image segmentation and rendered only after they have been converted to some shape. The key to this is our shape classifier, which we introduce to automatically decide which one of several fitted shapes "best" represents a region in an image segmentation. We designed our classifier specifically for use within NPR, it accepts arbitrary regions as input and returns a fitted shape chosen from one of a set of qualitative shapes such as ellipse, rectangle, or triangle.

The classifier allow us to emulate works by artists like Kandinsky and the later works of Matisse, as typified by his paper cut-outs. These artists transform complex geometric shapes into much simpler forms, typically circles, squares or triangles, or else shapes resembling convex hulls, for example. Segments also allow us to move away from stroke based NPR, so that we can make pictures from cloth, paper, marble blocks, wood, *etc.*

Broadly, our system works as follows. First we segment an image to obtain a hierarchy of regions, with small segments at the leaves of the tree and larger segments towards the root. Next we optimally fit several simple geometric shapes such as rectangles, triangles, circles, convex hull and so on to each segment. The classifier chooses the 'best' geometric shape from amongst those fitted. Hence, a generally circular segment is represented by a circle, a more-or-less square segment is given a square and so on. The convex hull is used whenever the segment has no particular shape. A supervised training paradigm is adopted for classifier generation, allowing for differences in user preference. Finally the geometry shapes are rendered, largest first so as to preserve salient detail. The system is able to work automatically using a default set of rendering parameters, though user interaction is also possible.

2 Background

The aim of NPR from images is to render real world photographs into artwork. This is a complex problem for many reasons, not limited to the diversity of media, of substrates, of styles of art work. The underlying problem is image understanding: the more an agent is able to recognise and understand image content, the more versatile it is when rendering. This is because creating artwork is, at root, a process of abstraction in which salient elements of a scene are given greater emphasis than non-salient elements. Unfortunately, image

- *Y.-Z. Song is with School of Electronic Engineering and Computer Science Queen Mary, University of London, London E1 4NS. E-mail: yzs@eecs.qmul.ac.uk*
- *D. Pickup, C. Li and P. Hall are with MTRC, Department of Computer Science, University of Bath, Bath, BA2 7AY. E-mail: {d.pickup, cl249, pmh}@cs.bath.ac.uk*
- *P. Rosin is with School of Computer Science & Informatics, University of Cardiff, Cardiff, UK. E-mail: Paul.Rosin@cs.cardiff.ac.uk*

understanding is such a complex problem that there has been no satisfactory solution to date, and none is likely in the near future. Nonetheless, it is possible to create high quality NPR from images, as this potted history reflects.

NPR from images can be traced back to the semi-automated paint systems of the early nineties [1], [2] which construct artwork as a sequence of virtual brush strokes. These influential papers side-stepped the question of image understanding by including the user — a trend still in evidence within the many interactive systems that exist today.

Media emulation is a natural consideration and has been with NPR from the early years [3]. The literature has produced algorithms capable of rendering photographs into oil paintings [4], cross hatchings [5], [6], and watercolors [7], to name but a few. Interest in emulating media continues [8], [9]. One of the most important questions any stoke based NPR system that strives for full automation must answer is: where should strokes be placed? Hertzmann goes some way to answer this by rendering strokes, at multiple scales, along contrast edges in an image [10], others use genetic search to direct stroke placement towards a salience map [11]. Media emulation is tangential to the concerns of this paper: we are happy to use media emulation during rendering, but our focus is an abstraction that is independent of media, as our results (Section 8) show.

Filtering is related to media emulation in that strokes are replaced by filtered pixel values. Filtering, though, inevitably relates directly to pixel values whereas media emulation can be wholly interactive, with no underlying picture necessary (the user is given a paintbox). Filtering methods are low level, meaning the methods assume little about image content and are therefore widely applicable. This is a very attractive characteristic, and the approach has a proven track record in producing high quality NPR from images [12], [13], [14], [15].

The purpose of filtering can be thought of as enhancing important detail. Similar to the problem facing automatic stroke placement, filtering algorithms must decide which areas of an image are important. Low-level image processes such as edge-maps [4] or color variance maps [16] have been used, and modern filtering methods have been developed that are "edge aware" [15], [17]. However, stroke placement and filtering are both improved if their outputs depend on information gathered from across the whole image, rather than a local image patch. This is in line with the idea that artworks are abstractions of the visual world: the importance of any one component depends on what other components are present. The typical approach is to use salience maps. Some use sophisticated interaction like eye tracking [18], others use salience maps that are automatically computed [11], [19], [20] so that an automatic painting system can focus on important areas of an image.

Segmentation is of prime concern here, its value in extending the gamut of NPR to abstract styles was recognized early [21] . The purpose of image segmentation is to partition an image into its semantic components, with no knowledge of what those components are. There is no segmentation algorithm that matches human performance, but state of the art is impressive nonetheless (*e.g.* [22], [23]. The value of segmentation to NPR from images is that it provides a *mid-level* model of the image to work with. So rather than having to deal with small patches of image that are essentially meaningless, the renderer deals with regions that have at least a weak semantic attribute, such as "this region is colour coherent".

For example, Mould [24], and later Setlur et al. [25] synthesize stained glass renderings by generating translucent texture patches, driven by a region segmentation. In the latter case, image content within each region determines the appearance of glass shards by visually querying a texture database. Collomosse and Hall [26] cut-out, re-arranged and distorted segmented image regions to create Cubist-like renderings from photographs. Xu *et al* use segments to produce Chinese paper cut-out art [27], while Orchard *et al* produce mosaics [28]. By replacing regions of an image with whole objects, Huang *et al* are able to produce Arcimboldo-like collages [29]. Finally, video segmentation is necessary to stabilise strokes when they are painted over moving objects [30], [31]. Mould provides an excellent summary of the use of regions in NPR from images [32].

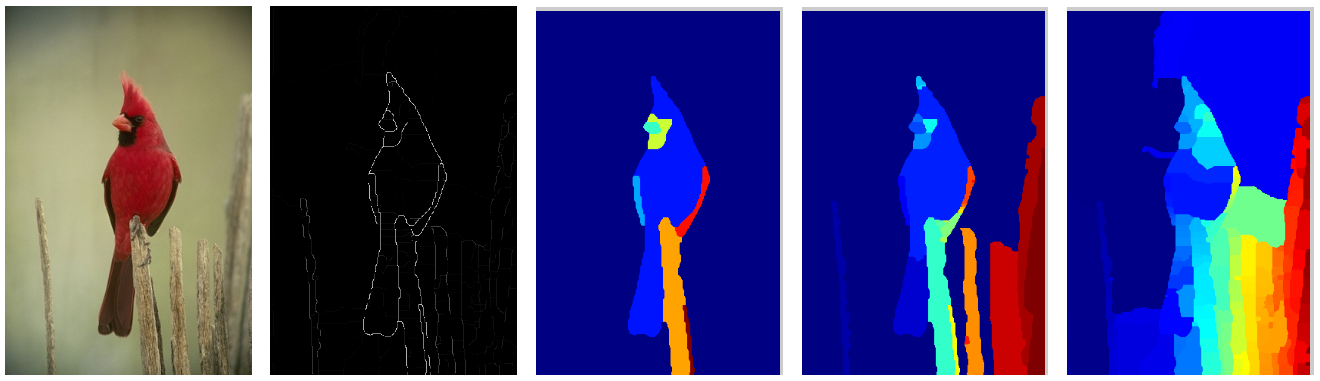
We classify segmented regions into one of a few shapes, which allows us to develop NPR towards the abstraction of artists such as Matisse or Kadinsky. Aside from this prime reason, we also take advantage of regions while rendering: in addition to painting regions with strokes we use textures to create marquetry images, marble mosaics, and cloth "stick ons". (see Section 8 for examples)

3 Overview

Our method is simple at heart: (i) build a segmentation hierarchy, (ii) decide the shape of the segments, then (iii) render the image using the shapes. Deciding the shape of a segment has two sub-steps: fit a set of candidate simple shapes, then choose the best amongst them.

Section 4 describes how we use a state of the art method to segment images into a set of layers, from fine grained (small segments) to coarse grained (large segments).

We optimally fit various shapes (rectangles, triangles, circles, superellipses, and a convex hull) to each segment, as described in Section 5. Which one of these shapes is used to represent the segment is decided by a classifier. As described in Section 6, our classifier makes an optimal classification by learning from users. Users choose which simple shape they believe best fits a randomly selected segment. Using a supervised training paradigm here is important since users often have different judgements towards what is the "best shape" that fits a region. Once it has been trained, the classifier is fully automatic.



(a) original image (b) Unfiltered UCM hierarchy [22] (c) Level 1 of filtered UCM hierarchy [23] (d) Level 4 of filtered UCM hierarchy [23] (e) Level 8 of filtered UCM hierarchy [23]

Fig. 1. Original image, its unfiltered hierarchical image description (98 levels in total), and three representative levels of the filtered hierarchy (8 levels in total)

The classifier assigns a simple shape to every segment in each layer. The shapes in the layers can then be rendered. There are many different rendering options for segments, Section 7 provides some examples.

4 Segmentation Hierarchy

Our approach to NPR is underpinned by automatic image segmentation. The aim of segmentation is to break an image into its meaningful parts such as 'dog', 'car', 'tree' and so on. This problem is beyond the scope of contemporary computer science; but significant progress has been made in recent years, enough to be of value to NPR. Among the many available segmentation algorithms, normalized cut and its variants [33], [34] have been the most successful. Recently the algorithm proposed by Arbelaez *et al* [22] is recognized as providing state of the art performance, using accepted measures to compare to human segmentations over the Berkeley database of images.

Conventional segmentation algorithms often yield interpretations of single scale, hence producing regions that are either entirely large or small – an artefact that is not desirable in our particular application setting (i.e., NPR). It is commonly acknowledged that artworks often exhibit an appropriate mixture of visual contents at multiple scales and various levels of saliency. Therefore, in this paper we aim to find regions from different segmentation scales, so to preserve an appropriate amount of detail, while keeping the abstractness. In that respect, we utilize hierarchical image segmentations instead, where segmentations from multiple scales are encapsulated.

Aside from its proven performance, Arbelaez *et al* [22] conveniently yields a hierarchical image segmentation. The hierarchy is a tree in which the topmost layers house coarse, containing large segments. The lowest layers are fine grained, having small segments that are children of the segments in the upper layers. Moreover, child segments are wholly contained by their single parent; this is not necessary for us but is a useful property

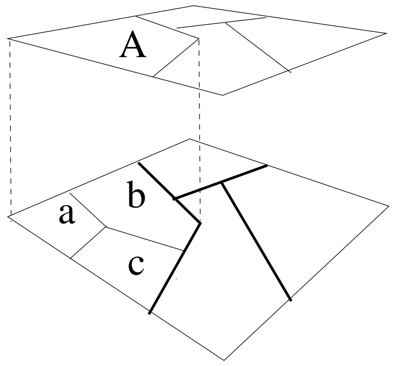


Fig. 2. A schematic diagram of a hierarchical segmentation. Upper layers hold large, coarse-scale segments such as A, which break into small segments in child layers such as a, b, and c. Each layer partitions the whole image. In a UCM for this example the dark edge would have value 2 (it appears in 2 layers) but the light edges would have value 1 (they appear in one layer only).

because smaller regions (*e.g.* eyes) are contained inside larger regions (*e.g.* faces). Figure 1(b) shows the weighted contour map of Figure 1(a)'s hierarchy, containing a total number of 98 levels.

This hierarchy is neatly contained inside a single image called an *Ultrametric Contour Map*, (UCM) [22]. An UCM is similar to an edge map in that is marks boundaries between objects, but there are two significant differences. (i) An UCM can be thresholded to create a segmentation, with different thresholds creating different levels. (ii) The UCM edges are computed not just between luminance gradients but between color and texture gradients too [35]. A UCM can contain hundreds of levels, which makes using it difficult. A UCM is schematically depicted in Figure 2.

We filter the hierarchy to create a new UCM, which reduces the number of levels by an order of magnitude [23]. Thus we generally deal with about 15 levels of hierarchy rather than about 200. This filtering process is meaningful as it yields segmentation hierarchies of manageable sizes, hence important later in rendering and user interaction. Figure 1(c)-(d) exhibit three representative levels (top, middle and bottom) of the filtered hierarchy (8 levels). It can be observed that the filtered hierarchy tends to have a helpful coarse to fine structure: the top level (Figure 1(c)) has the body of the bird segmented as a whole, while the tail among others were singled out in the middle level segmentation (Figure 1(d)) and the bottom level (Figure 1(e)) tends to have both the bird and background over-segmented.

Our approach depends on image segmentation and shape classification, so it might be argued that the quality of these algorithmic components limit the aesthetic value of our output. However, this argument carries weight only when the aim of image production is to maintain some semblance of recognisable parts in the output, because then the segmentation algorithm must yield semantic regions, and the classifier must produce sensible shapes. However, if total abstraction is the aim then the argument carries less weight, because abstract pictures do not have to possess any direct correspondence to the source.

5 Fitting Simple Shapes to Regions

Having segmented an image, we are able to fit a selection of shapes to each region. Specifically, we fit five shapes: circles, rectangles, triangles, superellipses and a "robust" version of the convex hull. Choosing the best shape from amongst these alternatives is the role of our classifier, and so far as we know is unique in shape fitting. Here we describe only how we fit each of the *specific* shapes mentioned to an image region, and the classifier that selects the best shape for each region is introduced in the next section.

Voss and Süße described a powerful method for fitting a variety of geometric primitives by the method of moments [36]. The particular primitive fitted is selected by the user. The data (the boundary of a segment) is first normalized by applying an appropriate transformation to put it into a canonical frame. The fitted geometric primitive is then simply obtained by taking the geometric primitive in the canonical frame and applying the inverse transformation to it. We have applied this approach to fit circles, rectangles and triangles. Let the number of parameters describing a geometric primitive be m and the number of transformation parameters be n; it is these parameters that are user specified. In the cases of circles and triangles the similarity ($n = 4$) and affine ($n = 6$) transformations are used respectively. Since in both cases $m \leq n$ the geometric primitive can be found directly. In the case of the rectangle ($m = 5$) there is no general transformation group for the class of all rectangles. Therefore, Voss and Süße use a similarity transformation, and so an $m - n = 1$D optimization is required to completely determine the fit [37].

As an example, further details of rectangle fitting are given. The standard rectangle in the canonical frame is taken to be centred at the origin and aligned with the axes. Its area is set to a, and so the coordinates of its vertices are $(\pm\frac{1}{2}\sqrt{a}, \pm\frac{1}{2}\sqrt{a})$. The normalization steps for fitting a rectangle are as follows, where the goal is to normalize the data such that its geometric moments m_{ij} match those of the canonical rectangle and become $m_{00} = 1, m_{10} = 0, m_{01} = 0, m_{11} = 0$:

1) Normalise the position by translating the data by $(-m_{10}, -m_{01})$ so that its centroid is at the origin.
2) Normalise the orientation by a rotation of $-\phi$ where $\tan 2\phi = \frac{2\mu_{11}}{\mu_{20}-\mu_{02}}$ and μ_{ij} are the central moments of the data.
3) Perform scaling in X and Y by $\frac{1}{\sqrt{a}}$. Since a is unknown it is determined by a least squares optimization between the following moments of the standard rectangle – $M_{20}(a) = \frac{1}{12}a, M_{02}(a) = \frac{1}{12a}, M_{40}(a) = \frac{1}{80}a^2, M_{22}(a) = \frac{1}{144}, M_{04}(a) = \frac{1}{80a^2}$ – and the corresponding moments of the normalized data.

To fit superellipses a closed form solution is not available using the above approach. A superellipse centred on the origin with axes aligned with the co-ordinate system can be represented by the following implicit equation:

$$\left(\frac{x}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon}} = 1.$$

The ideal distance measure to minimise would be the shortest Euclidean distances between each data point and the superellipse, but this is expensive to compute. An alternative commonly used in the fitting literature is the algebraic distance, which for the superellipse is given by

$$Q(x,y) = \left(\frac{x}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon}} - 1.$$

Although simple and efficient to use, the algebraic distance has the disadvantage of a *curvature bias*; that is, it produces relative underestimates near high curvature sections of the superellipse compared to low curvature sections. Since errors are underestimated near high curvature sections then these data points will have less effect on the fitting than other points, which tends to cause the eccentricity of the fitted superellipse to be overestimated.

Instead, we follow Rosin and West [38] who provide a simplification of the shortest Euclidean distance that avoided the curvature bias. For a superellipse centred at the origin and aligned with the co-ordinate axes, they minimised the Euclidean distance d_p from the data point (x_p, y_p) to the point (x_s, y_s) on the superellipse along the ray that passes through (x_p, y_p) and the centre of the superellipse $(0, 0)$, where:

$$x_s = \left| \frac{1}{\frac{1}{a^{\frac{2}{\varepsilon}}} + \left(\frac{y_p}{x_p b}\right)^{\frac{2}{\varepsilon}}} \right|^{\frac{\varepsilon}{2}}$$
$$y_s = x_s \frac{y_p}{x_p}$$

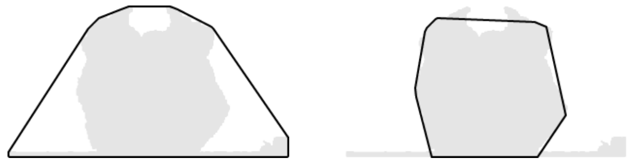


Fig. 3. Left: Standard convex hull fitted to region; Right: Robust convex hull fitted to region

To avoid evaluating complex roots for negative values, the symmetry of the superellipse allows x_p and y_p to be replaced by their absolute values.

The summed distances along these rays are minimized using Powell's method for non-linear optimization [39] which just requires the term being minimized to be evaluated. The optimization is initialized by fitting an ellipse to the data. A superellipse not in the canonical position and orientation can be described as

$$\left(\frac{(x-x_c)\cos\theta-(y-y_c)\sin\theta}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{(x-x_c)\sin\theta+(y-y_c)\cos\theta}{a}\right)^{\frac{2}{\varepsilon}}$$

but since it is not longer symmetric in all four quadrants, complex roots cannot be avoided by simply using absolute values. Therefore optimization is performed by inversely transforming the data rather than transforming the superellipse model which is kept in its canonical position.

The convex hull is an attractive symbolic representation of a shape on two counts. It is generally more compact (using only a subset of the original polygonal vertices), and also perceptually simpler since all indentations have been removed. However it has two limitations: it is insensitive to the size and shape of all indentations, and is also too sensitive to protrusions.

To overcome these problems Rosin and Mumford [40] suggested a "robust" version of the convex hull, which is the convex polygon that maximises the area overlap with the input polygon. To compute the robust convex hull they used a genetic algorithm which is the approach followed here (alternatively a dynamic programming solution has been described [41]). The genetic algorithm represents solutions as bit-string chromosomes in which a subset of the vertices of each each region boundary are selected. These chromosomes are "repaired" to make them valid solutions by replacing them with their (standard) convex hulls. The quality of an individual is determined by its overlap with the original boundary; this is calculated by computing the exclusive OR operation of the two polygons. A standard steady-state genetic algorithm with mutation and one-point crossover is then applied to find the best individual. An example is shown in Figure 3 in which a region is fitted with the standard and robust convex hulls. While the mis-segmentation at the bottom has had a large effect on the standard convex hull the robust convex hull is hardly effected, and provides a much better representation of the shape.

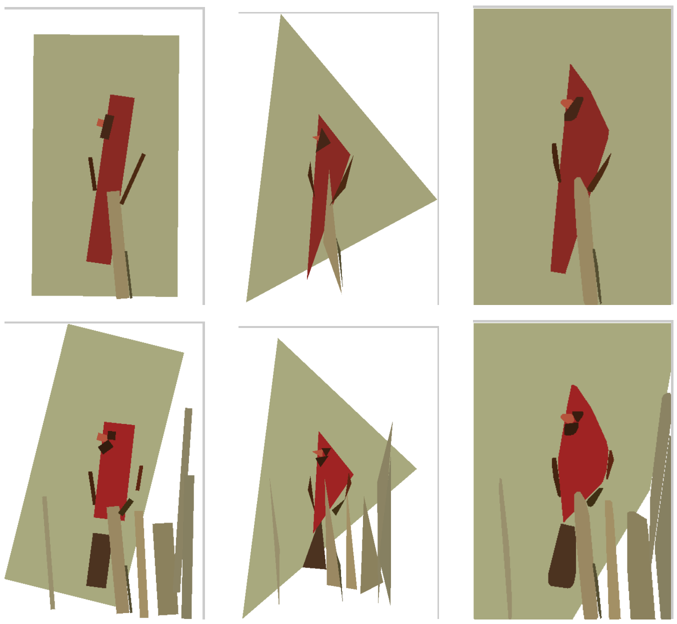


Fig. 4. Results of fitting shapes of a single type to Level 1 and 4 respectively. Left Column: Fitting Rectangles; Middle Column: Fitting Triangles; Right Column: Fitting Robust Convex Hulls

Results of fitting various user defined shapes to the two levels in the segmentation hierarchy shown in Figure 1(c) 1(d) are given in Figure 4.

6 Automatic Shape Type Selection

Recall that in the abstract art of interest here objects are made up of a collection of simple shapes. In our case we must replace each region in a segmented image with one of the canonical shapes. This could be done interactively, or a forced choice could be made such as always choosing a robust convex hull; the first option is tedious for the artist — the second leads to tedious artwork for viewers. We therefore automatically select the best simple shape from amongst a set of canonical alternatives. To the best of our knowledge our *automatic shape selection* is unique and so is key to the contribution of this paper.

We considered several varieties of information theoretic approaches, to decide which canonical shapes was best. However, we found none corresponded well with subjective judgements of which qualitative shape to use. The situation is somewhat analogous to using root mean square error (RMS) to measure the quality of different image decompression algorithms: it is well known (if only anecdotally) that a small RMS error does not necessarily imply a high quality algorithm, and conversely that a high RMS error does not always imply a low quality algorithm. Figure 5 exhibits results of automatic shape selection using a simple RMS error. It can be seen that these selections tend to be relatively random and

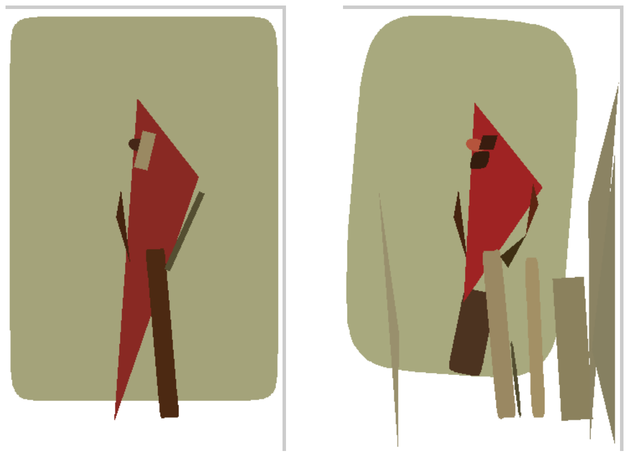


Fig. 5. Naive automatic shape selection results using a simple RMS error. Left to right: selection results of Level 1 and 4 respectively

even odd in some cases, such as the large superellipses fitted to the background, but for regions (such as the peck) that are more or less "perfect" geometric shapes (ellipse in this case), the fitting is rather accurate as expected. Perhaps a more important downside of using RMS error is that it doesn't allow subjectivity, a key element in art making, into the process.

With this in mind we opted in favor of a trained classifier. The fact that classifier is supervised allows considerable flexibility — many users can train the system so that the shapes chosen represent some kind of average, or a single user might wish to bias the classifier in favor of a particular shape. We now explain our classifier and how we trained it, starting with an overview before providing further details.

Automatically selecting appropriate shape models is done using a standard supervised classification paradigm. Specifically, we construct a C4.5 decision tree [42] through a learning process governed by a human user. In our case the user collects a set of training regions, then labels each region with the name of a canonical shape, such as triangle, rectangle, and so on. This human labeled data provides a ground truth upon which the classifier is constructed. To do this our system optimally fits each of the canonical shapes to each labeled region, and then gathers statistics about the fit. As explained below, these statistics are used as elements of a feature vector that characterize the fit of all canonical shapes to the labeled region. These labeled feature vectors are used by the system to build the C4.5 decision tree, as described in [42].

6.1 Features

As mentioned, each region is described by a feature vector. These features are the basis for making the decision regarding which is the most appropriate canonical shape. Our feature vectors comprise statistics that describe the probability distribution of the unsigned errors between the region and each of the fitted shape. We obtain a set of errors by choosing a set of points around the region boundary and measuring the shortest distance to the fitted shape. The shortest distance is determined using the distance transform. These errors are invariant to translation and rotation, and are further normalised for scale by dividing by the square root of the area of the region.

Using the summed error as a single value to describe the quality of fit for each geometric model is not sufficient to decide which canonical shape is best according to aesthetics and perceptual criteria. Indeed, it is easy to construct examples in which the best canonical shape does not have a lower summed error. Instead, the histogram distribution of unsigned errors is considered, and summarized by the following statistics: mean, μ, standard deviation, σ, skew ρ, and kurtosis κ. If there are N error measures, e_i, these are defined by

$$\mu = \frac{1}{N}\sum_{i=1}^{N}|e_i| \tag{1}$$

$$\sigma = \left[\frac{1}{N}\sum_{i=1} N(|e_i|-\mu)^2\right]^{1/2} \tag{2}$$

$$\rho = \frac{1}{\sigma^3}\sum_{i=1}^{N}(|e_i|-\mu)^3 \tag{3}$$

$$\kappa = \frac{1}{\sigma^4}\sum_{i=1} N(|e_i|-\mu)^4 - 3 \tag{4}$$

We collect $N = 4$ such numbers for each canonical shape, and consider five canonical shapes. Thus each region is associated with twenty numbers

$$f = [\mu_1, \sigma_1, \rho_1, \kappa_1, \mu_2, \sigma_2, ...\kappa_M], \tag{5}$$

which is its feature vector.

Figure 6 shows an example of fitting the canonical shapes to a region. It can be seen that the rectangle, superellipse and robust convex hull provide better fits than the circle and triangle, and this is reflected in the distribution of their residual errors. In particular, the errors are packed close to zero in 'good fits' whereas the errors are more evenly distributed for 'poor fits'.

6.2 The classifier

The basis of a C.45 decision tree is that each feature can be used to make a decision that splits the data into smaller subsets. This can be regarded as partitioning feature space into equivalence classes using axis-parallel hyperplanes. The method places the most informative feature closer to the root of the tree; the degree of information is measured by an entropy based measure, namely Normalized Information Gain. Given a set S of items, each with a class label drawn from the set of M classes C_i, then the entropy of S is defined to be

$$I(S) = -\sum_{i=1}^{M} F(C_i, S)\log F(C_i, S)$$

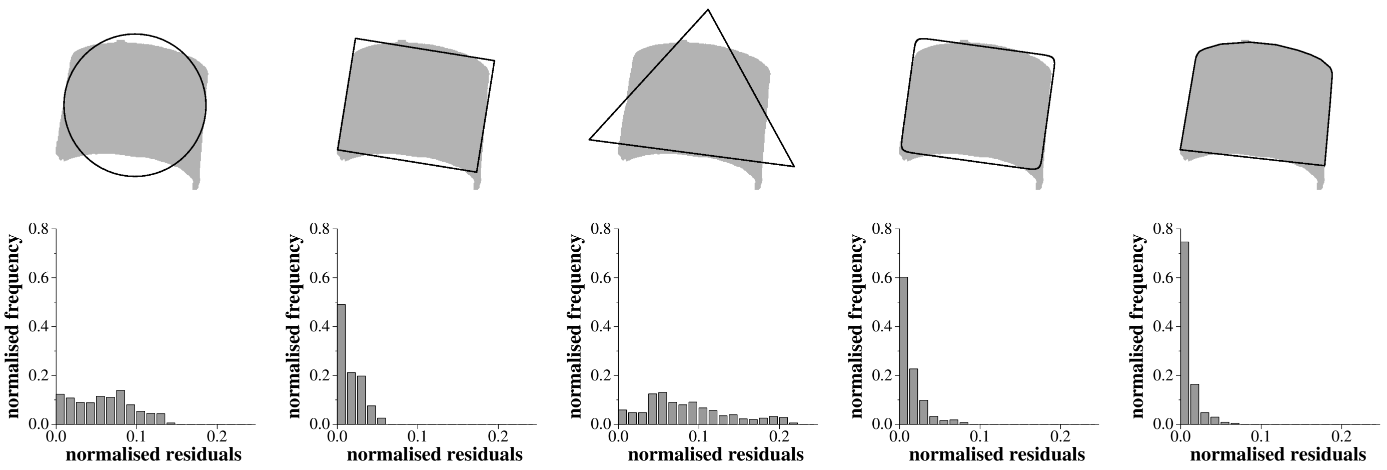


Fig. 6. Above: shapes fitted to a region (circle, rectangle, triangle, superellipse and robust convex hull). Below: histograms of normalised residual errors for the region and the above shapes.

where $F(C_i, S)$ is the relative frequency of items in S with class label C_i. If the set of items S is partitioned into m subsets $S_1, S_2, \ldots S_m$, then the Normalized Information Gain is calculated as

$$\mathrm{NIG}(\{S_i\}_1^m) = \frac{\left(I(S) - \sum_{i=1}^m \frac{|S_i|}{|S|} I(S_i)\right)}{\left(-\sum_{i=1}^m \frac{|S_i|}{|S|} \log \frac{|S_i|}{|S|}\right)}.$$

For numerical features (as used in this paper) the decision at a node in the decision tree is of the form *feature value* $\leq$ *threshold*. To calculate the effectiveness of a feature, all instances in S are sorted, and the mean of each adjacent pair of features determines a threshold value. The threshold maximising Normalized Information Gain is selected. The data is recursively partitioned using a divide and conquer algorithm until no improvement in the classification of the training data is possible. However, this generally leads to overfitting, and so a subsequent pruning stage is applied to the tree in which subtrees that provide little predictive accuracy are replaced by leaves.

6.3 Classifier training

Following this approach, we selected two training images not used in this paper (shown in Figure 7). We segmented them using the segmenter described in Section 4. The different granularities used provided a large variety of over 100 regions. A feature vector is then computed for each one, and training data is generated by ascribing the labels with the name of 'superellipse', 'circle', 'triangle', and 'rectangle' only, with 'convex hull' being the default label.

As previously mentioned, a key property of the proposed supervised training approach to automatic shape selection lies with the degree of flexibility it offers. It is important to recognise that the shape classifier is trained by a user, consequently the 'best' shape chosen can have a subjective quality. The consequence of training is that it allows for both (i) aesthetic considerations to be incorporated, overriding fitting errors, and (ii) allows the user to customise the classifier so that different users can generate different styles — or a single user may want to be able to generate multiple styles.

Fig. 7. The two training images used in this paper

Figure 8 illustrates three classifiers each trained by a different user. It shows that user preferences can be integrated. Classifier 1 is trained by someone more con-

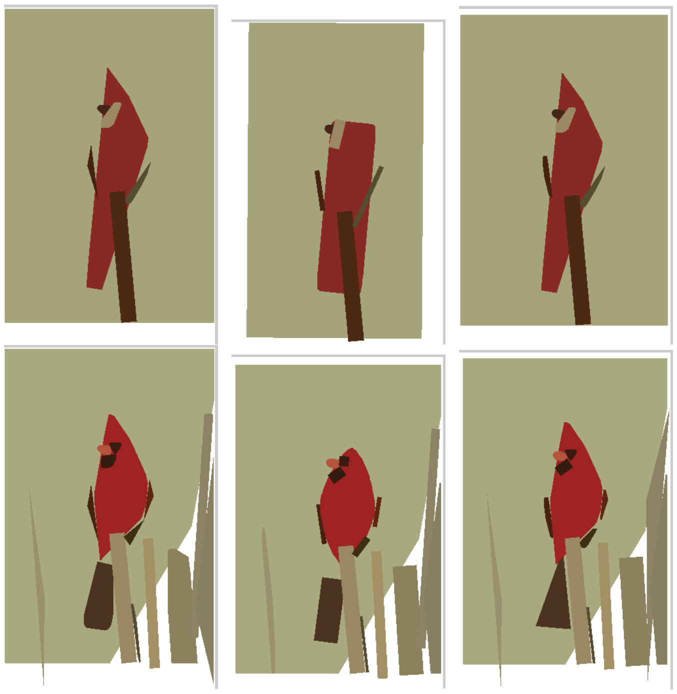


Fig. 8. Comparison of renderings (Level 1 and 4) from three classifiers. Left to right (column-wise): classifier 1, classifier 2 and classifier 3 (default used to produce all renderings in this paper)

servative, in that more shapes were labelled as 'convex hulls' being the most 'accurate' fits. Classifier 2 on the other hand, offers the most abstraction among the three – the user used more 'superellipses' and 'rectangles' in his training set. The third classifier offers a more balanced selection of shapes and is the default one used throughout this paper. It is important to note that no 'best' classifier exists, since users often disagree with each other towards the assignment of shape labels, or one might just be feeling adventurous and artistic on the day. Overall, this training process implicitly addresses differences in stylisation. To further analyse the aesthetics and other perceptual differences between the learnt stylisations would require user studies, and is outside the scope of this work.

7 Rendering Shapes

We can now fit shapes to each region from a given segmentation and automatically select the best fit among those. Segmentations at a coarser level can yield large and more abstract shapes; whereas, detailed segmentations often result in shapes that are too small and overly detailed. What we really want is to preserve an appropriate amount of details, while keeping the abstractness. In this paper, for a segmentation hierarchy of N levels, we found that using the first level and the middle level (i.e., N/2) produces renderings of appropriate abstraction. However, such appropriateness is subject to the artistic judgements of the authors. We allow for user interaction to accommodate subjectivity into the process, as explained towards the end of this section.

7.1 Level-of-detail control

However, simply overlaying shapes from the top layer onto the bottom layer will produce renderings of limited abstraction – as the previous layer will be covered by the next layer. We resolve this issue by treating the top layer of larger/coarser shapes as "background" and the bottom one with smaller and detailed shapes as "foreground". We then filter the detailed shapes on the top layer by their corresponding shapes underneath. More specifically, we only render shapes from the top layer whose color deviates from that of the shape underneath above a certain threshold. Hertzmann also used color differences to place strokes on top of those already painted in his stroke rendering work [10]. Unlike him, we measure color differences in terms of just noticeable difference (jnd) in CIELAB color space. For instance, take two colors, (L_1, a_1, b_1) and (L_2, a_2, b_2), we define their color difference ΔE_{12} as follows

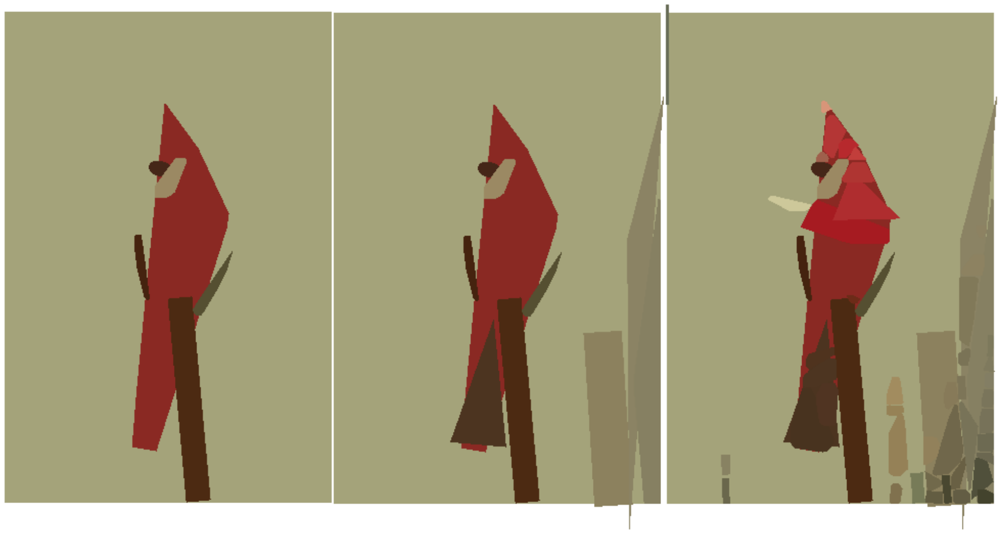
$$\Delta E_{12} = \frac{\sqrt{(L_1 - L_2)^2 + (a_1 - a_2)^2 + (b_1 - b_2)^2}}{jnd}$$

where $jnd \approx 2.3$ in CIELAB color space [43]. Therefore, in general, ΔE measures how many jnds one color deviates from another. By placing a threshold on ΔE, we can control the level of detail to render on the top layer; increasing the threshold results in fewer shapes being rendered and vice versa. For all results in this paper, a constant threshold of 5 is used on ΔE.

When it comes to rendering shapes into a framebuffer, the ordering of shapes can play a role in the final output as well. This is because of the fact that shapes fitted often overlap each other and which one comes on top can confuse the viewer's perception. We tackled this problem by introducing a shape fitting error τ. Shapes with large fitting errors are rendered before those with smaller errors. Given a shape model s and its corresponding region r, we denote the area of s by S and similarly R for r, then τ is defined as the following ratio, $|\bigcap(S,R)|/|\bigcup(S,R)|$, which is a form of Tanimoto similarity score, calculated on a per pixel basis. The idea is then to lay down shapes according to their assigned fitting error. In practice small regions tend to produce smaller fitting errors, hence they are likely to be laid after large regions as details.

7.2 User interaction

We optionally allow the user to edit the rendering in two ways. Firstly users are able to change the level of detail by altering which layers of the hierarchy are used. Figure 9(a) shows the same image, but rendered with increasing levels of detail. Secondly the user has control over the individual shapes in the output image. Any of the shapes can be selected and either deleted or altered by changing the colour, rendering style, or


(a) Rendering using different levels in the segmentation hierarchy. From left to right, first level only, middle level on top of first level and the bottom level on top of both first and middle levels.

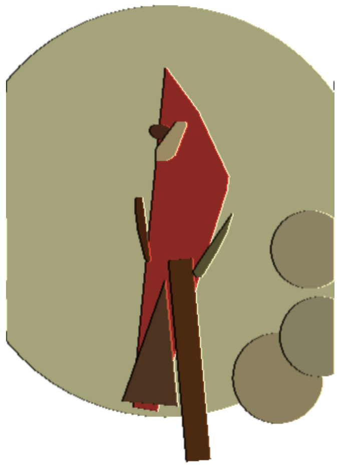

(b) An example rendering after user interaction

Fig. 9. Rendering detail control

shape type fitted to the region (circle, triangle, ect..). Figure 9(b) offers a "paper cut-outs" rendering where the background is abstracted as a collection of circles. More examples of renderings via user interaction are exhibited in the next section, together with a collection of automatically generated ones.

8 Results: A Gallery of Renderings

This section exhibits a collection of shape rendering results. Here we only show a few of the many possible media emulations, we could have included watercolor, stained glass, mosaics, Chinese paper cut-outs or any of the many techniques that have been developed in NPR over the years.

We begin with two renderings of the "bird" image used throughout the paper shown in Figure 10. No user interaction was performed in both renderings, but only styles were altered. Top of the figure offers a "paper cut-outs" style rendering, whereas the bottom rendering emulates graffiti art.

Figure 11 demonstrates how simple user interaction can help to produce better renderings. Original rendering is shown in the middle, where the final rendering (bottom) has all background shapes removed.

Another set of renderings are shown in Figure 12. It shows a source photograph and two outputs from our system, one automatically produced (middle) the other with user editing (bottom). The rendering in the middle exhibits the "paper cut-outs" style and the bottom one rendered in "cartoon". Color editing was performed on the bottom rendering to create a very different look: the large square in the background was changed to blue, the colour of a green convex hull near the top left was changed to grey and that of a convex hull near the top left was changed to yellow.

Figure 13 offers another set of renderings, middle one in "paper cut-outs" and bottom in wood mosaic. The rendering in the middle is automatically produced, whereas the bottom one was done under user interaction. Different to the previous example where shape colours were altered, we only change shapes in this example. In particular, all the shapes which form part of the car have been left unchanged, all others have been set to squares, resulting in fourty shapes changed in total.

Figure 14 exhibits another set of renderings with user edits in all aspects allowed, i.e., hierarchy, shape type/color, and style. The "tree" in Figure 14 was abstracted as a circle as opposed to a "convex hull" and its shadow changed to a rectangle, colors of the top three shapes in the sky were also changed to boost aesthetic quality.

Finally, we conclude the gallery with a rendering of the famous "Dora Maar" painting by Pablo Piccaso, shown in Figure 15.

It is worth noting that our method relies on state-of-the-art image segmentation techniques to produce a hierarchy of regions. However, despite the many exciting advances to-date, image segmentation is still largely an unsolved problem. For example, it might be subjectively argued that the top layer of the "plane" in Figure 12 is over-segmented, hence producing a number of undesirable shapes overlaying the bottom layer (e.g., on the body of the plane), and the surplus region appearing to the right and under the tree in Figure 14 also seemed odd. Solving the segmentation problem is beyond the scope of this paper, alternatively we allow user interaction in the process to facilitate the removal of these artefacts. One interesting future work would be increasing the flexibility of user interaction. For example, individual shapes should not only be removable, but also addable to the present rendering based on user

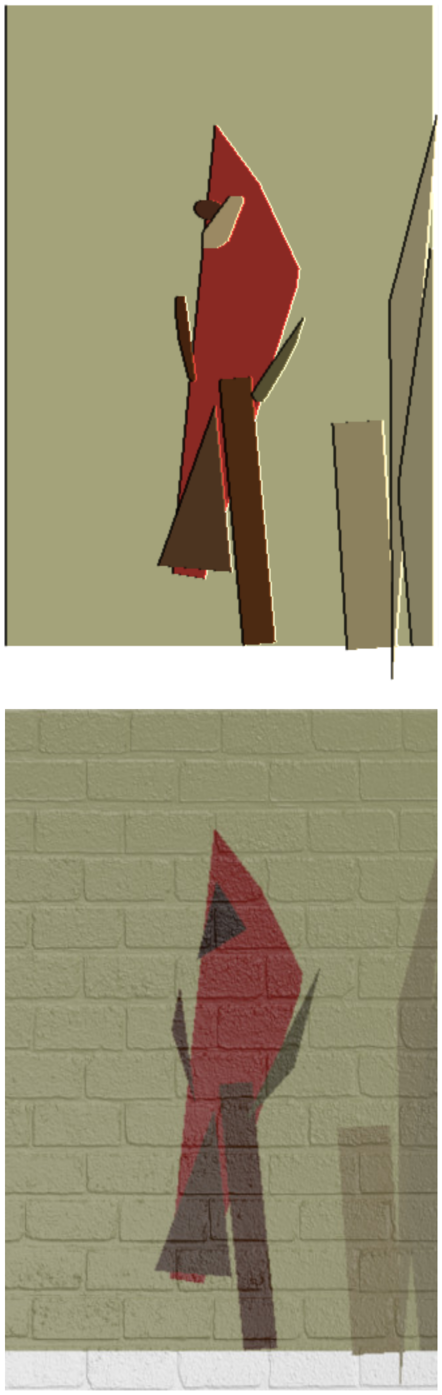


Fig. 10. Two renderings of the "bird" image

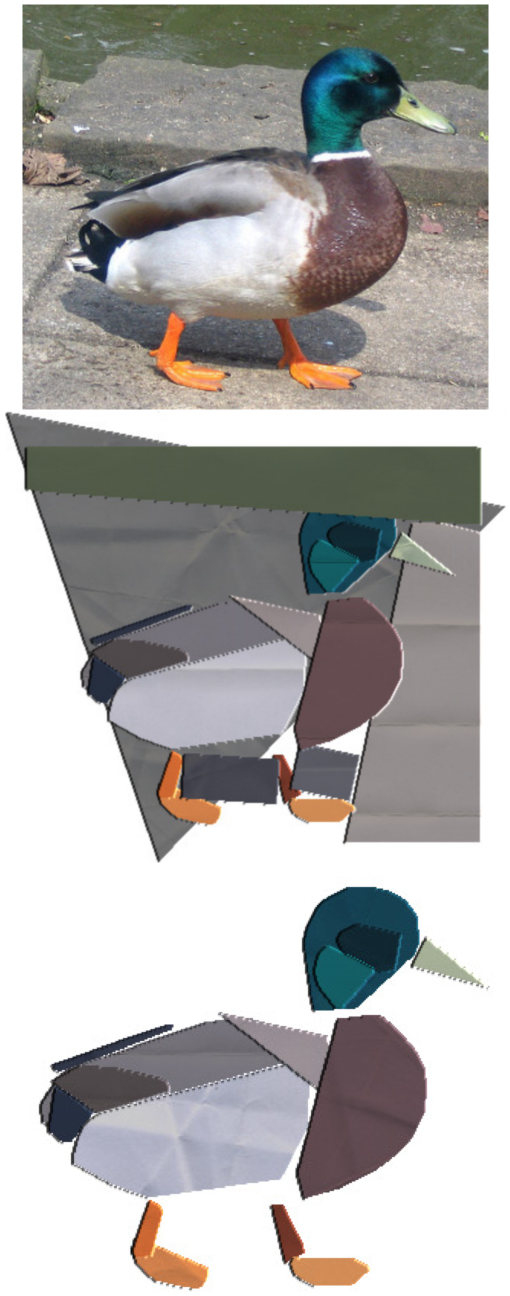


Fig. 11. From top to bottom: original image of a duck, an automatic "paper cut-outs" rendering, and one after user removal of the background

feedbacks such as mouse clicking.

9 Conclusion

This paper's contribution to NPR is that we have moved towards automatically creating more abstract art than was previously possible. More specifically, the art we synthesized was influenced by artists such as Kandinsky and later Mattise who advocated the use of geometric shapes. Shape simplification is the key to delivering the level of abstraction resembled in such type of art. Importantly, we can automatically select which shape fits the best among a few that we can fit. We are also able to combine shapes of various granularities and so preserve appropriate amount of detail.

The automatic shape selection step involves supervised learning and the classifier can be re-used once trained. No further input is required unless one wishes to paint with an advanced media style — but control of that part lies rightfully with other literature. In the gallery of Section 8 we exhibit direct output that is similar in abstraction to Kandinsky et al, a cut-paper

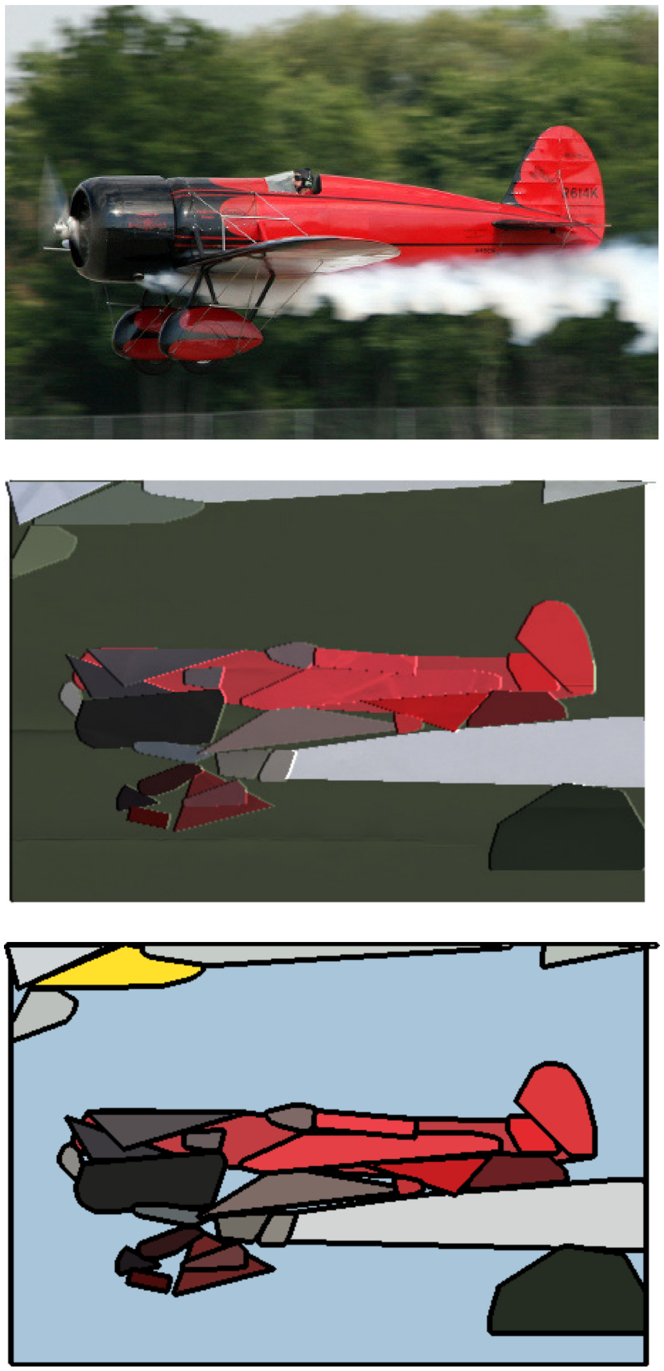


Fig. 12. Original image of a plane, and two artistic renderings

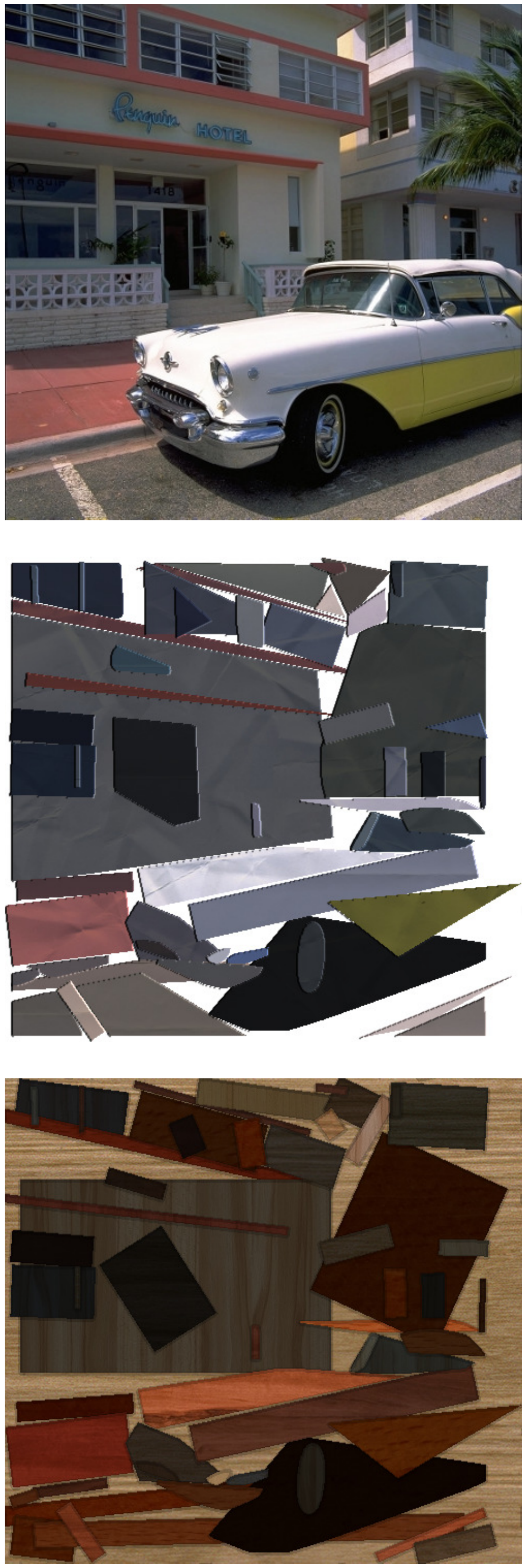


Fig. 13. Original image of a street scene in Miami, and two artistic renderings

effect as used by Matisse in his later works, as well a more traditional media of oil and crayon paintings.

In conclusion, it is clear that shape simplification is able to support the creation of synthetic artworks of a more abstract nature than before. There is plenty of scope for future work in this area.

Acknowledgments

We would like to thank all reviewers for their comments that helped to make this paper a stronger contribution.

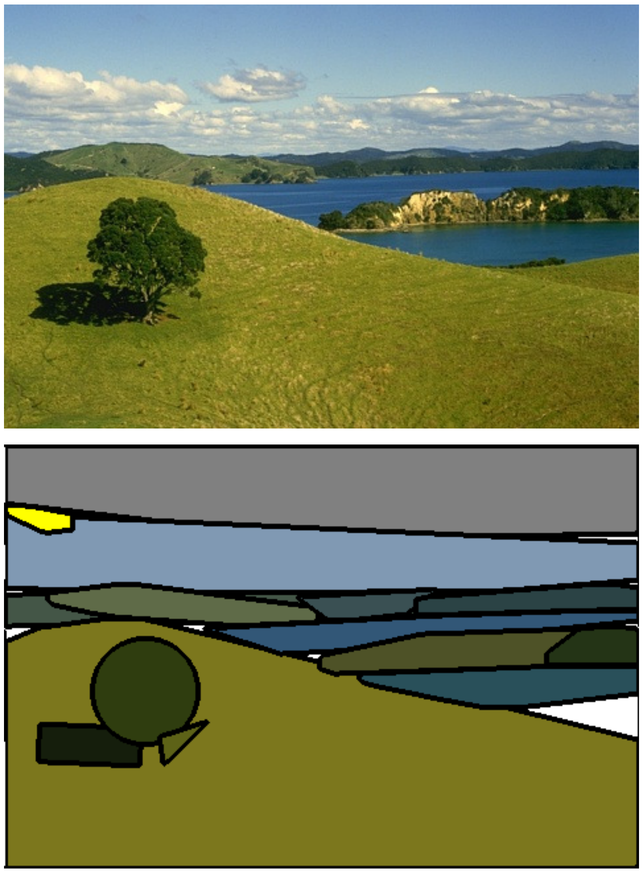


Fig. 14. Original image of a landscape scene, and a "cartoon" rendering

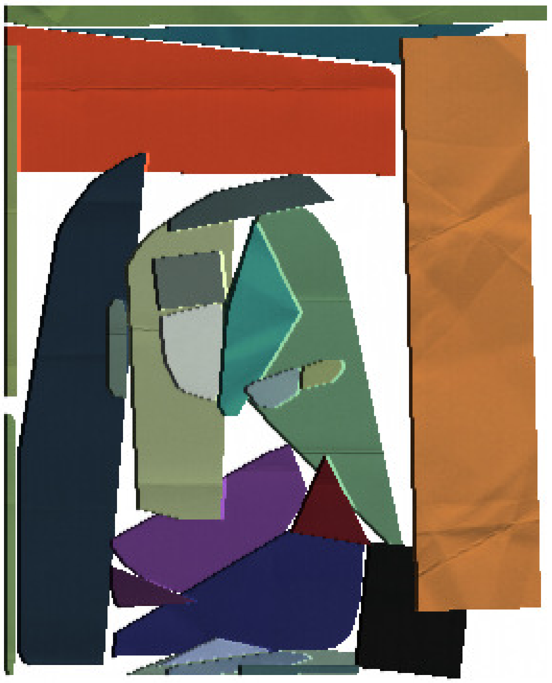


Fig. 15. A paper-cut rendering of the "Dora Maar" painting by Pablo Piccaso

REFERENCES

[1] P. Haeberli, "Paint by numbers: abstract image repre sentations," in *Proc. 17th Intl. Conference on Computer Gr aphics and Interactive Techniques (ACM SIGGRAPH)*, vol. 4, no. 24, 1990, pp. 207–214.

[2] M. P. Salisbury, S. E. A. an d Ronen Barzel, and D. H. Salesin, "Interactive pen-and-ink illustration," in *Proc. 21st Intl. Conference on Computer Gr aphics and Interactive Techniques (ACM SIGGRAPH)*, Florida, USA, 1994, pp. 101–108.

[3] T. Cockshott, J. Patterson, and D. England, "Modelling the texture of paint," *Computer Graphics Forum*, vol. 11, no. 3, pp. 217–226, 1992.

[4] P. Litwinowicz, "Processing images and video for an imp ressionist effect," in *Proc. 24th Intl. Conference on Computer Gr aphics and Interactive Techniques (ACM SIGGRAPH)*, Los Angeles, USA, 1997, pp. 407–414.

[5] M. P. Salisbury, M. T. Wong, J. F. Hughes, and D. H. Salesin, "Orientable textures for image-based pen-and-ink illustration," in *Proc. 24th Intl. Conference on Computer Graphics and Interactive Techniques (ACM SIGGRAPH)*, Los Angeles, USA, 1997, pp. 401–406.

[6] P. Hall, "Non-photorealistic rendering by Q–m apping," *Computer Graphics Forum*, vol. 1, no. 18, pp. 27–39, 1999.

[7] C. Curtis, S. Anderson, J. Seims, K. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proc. 24th Intl. Conference on Computer Gr aphics and Interactive Techniques (ACM SIGGRAPH)*, 1997, pp. 421–430.

[8] S. Brooks, "Mixed media painting and portraiture," *IEEE Trans. on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 1041–1054, 2007.

[9] R. Vergne, D. Vanderhaeghe, J. Chen, P. Barla, X. Granier, and C. Schlick, "Implicit brushes for stylized line-based rendering," in *Eurographics*, 2011.

[10] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proc. 25th Intl. Conference on Computer Gr aphics and Interactive Techniques (ACM SIGGRAPH)*, 1998, pp. 453–460.

[11] J. P. Collomosse and P. M. Hall, "Salience-adaptive painterly rendering using genetic search," *Intl. Journal on Artificial Intelligence Tools (IJAIT)*, vol. 15, no. 4, pp. 551–576, Aug. 2006.

[12] H. Winnemöller, S. Olson, and B. Gooch, "Real-time video abstraction," in *SIGGRAPH*, 2006, pp. 1221–1226.

[13] H. Kang, S. Lee, C.K., and Chui, "Flow-based image abstraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 62–76, 2009.

[14] P. L. Rosin and Y. K. Lai, "Towards artistic minimal rendering," in *Int. Symp. on Non-Photorealistic Animation and Rendering (NPAR)*, 2010, pp. 119–127.

[15] J. E. Kyprianidis and H. Kang, "Image and video abstraction by coherence-enhancing filtering," in *Eurographics*, 2011.

[16] S. M. F. Treavett and M. Chen, "Statistical techniques for the automated synthesis of non-photorealistic images," in *Proc. 15th Eurographics UK Conference*, 1997, pp. 201–210.

[17] S. Paris, S. W. Hasinoff, and J. Kautz, "Local laplacian filters: edge-aware image processing with a laplacian pyramid," in *ACM SIGGRAPH 2011 papers*, ser. SIGGRAPH '11. New York, NY, USA: ACM, 2011, pp. 68:1–68:12. [Online]. Available: http://doi.acm.org/10.1145/1964921.1964963

[18] D. DeCarlo and A. Santella, "Stylization and abstraction of photographs," *ACM Trans. Graph.*, vol. 21, pp. 769–776, July 2002. [Online]. Available: http://doi.acm.org/10.1145/566654.566650

[19] J. P. Collomosse and P. M. Hall, "Painterly rendering using image salience," in *Proc. 20th Eurographics UK Conference*, 2002, pp. 122–128.

[20] J. Bangham, S. Gibson, and R. Harvey, "The art of scale-space," in *British Machine Vision Association*, 2003, pp. 569–578.

[21] B. Gooch, G. Coombe, and P. Shirley, "Artistic vision: Painterly rendering u sing computer vision techniques," in *Proc. 2nd ACM Symposium on Non-photorealis tic Animation and Rendering*, Jun. 2002, pp. 83–90.

[22] P. A. amd M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *Computer Vision and Pattern Recognition*, 2009.

[23] Y.-Z. Song, P. Arbelaez, P. Hall, C. Li, and A. Balikai, "Finding semantic structures in image hierarchies using laplacian graph energy," in *Computer Vision – ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds., vol. 6314. Springer, 2010, pp. 694–707.

[24] D. Mould, "A stained glass image filter," in *Rendering Techniques*, 2003, pp. 20–25.

[25] V. Setlur and S. Wilkinson, "Automatic stained glass rendering," in *Proc. Computer Graphics Intl. (CGI)*, 2006, pp. 682–691.

[26] J. P. Collomosse and P. M. Hall, "Cubist style rendering from photographs," *IEEE Transactions on Visualization and Compute r Graphics (TVCG)*, vol. 4, no. 9, pp. 443–453, Oct. 2003.

[27] J. Xu and C. Kaplan, "Artistic thresholding," in *Non-photorealistic Animation and Rendering*, 2007, pp. 39–47.

[28] J. Orchard and C. Kaplan, "Cut-out image mosaics," in *Non-photorealistic Animation and Rendering*, 2008, pp. 79–87.

[29] H. Huang, L. Zhang, and H.-C. Zhang, "Arcimboldo-like collage using internet images," in *Proceedings of the 2011 SIGGRAPH Asia Conference*, ser. SA '11. New York, NY, USA: ACM, 2011, pp. 155:1–155:8. [Online]. Available: http://doi.acm.org/10.1145/2024156.2024189

[30] J. Wang, Y. Xu, H. Shum, and M. Cohen, "Video tooning," *ACM Transactions on Graphics*, vol. 3, no. 23, pp. 574–, 2004.

[31] D. R. Collomosse, John P. and P. M. Hall., "Stroke surfaces: Temporally coherent artistic animations from video." vol. 11, no. 5, 2005), pages = 540-549.

[32] D. Mould, "Region-based abstraction," pp. 125–148, 2012.

[33] J. Shi and J. Malik, "Normalized cuts and image segmentation," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, ser. CVPR '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 731–.

[34] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multi-scale graph decomposition," in *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1124–1131.

[35] D. Martin, C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color and texture cues," *IEEE TPAMI*, vol. 26, pp. 530–549, 2004.

[36] K. Voss and H. Süße, "Invariant fitting of planar objects by primitives," *IEEE Trans. on Patt. Anal. and Mach. Intell.*, vol. 19, no. 1, pp. 80–84, 1997.

[37] H. Süße and K. Voss, "A new efficient algorithm for fitting of rectangles and squares," in *Int. Conf. Image Processing*, 2001, pp. 809–812.

[38] P. L. Rosin and G. A. W. West, "Curve segmentation and representation by superellipses," *Proc. IEE: Vision, Image, and Signal Processing*, vol. 142, pp. 280–288, 1995.

[39] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vettering, *Numerical Recipes in C*. Cambridge University Press, 1990.

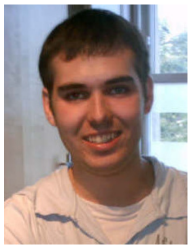
[40] P. L. Rosin and C. L. Mumford, "A symmetric convexity measure," *Computer Vision and Image Understanding*, vol. 103, no. 2, pp. 101–111, 2006.

[41] A. Kolesnikov and P. Fränti, "Optimal algorithm for convexity measure calculation," in *Int. Conf. Image Processing*, 2005, pp. 353–356.

[42] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[43] G. Sharma, *Digital Color Imaging Handbook*. CRC Press, 2003.

Yi-Zhe Song Yi-Zhe Song received both the B.Sc. (first class) and Ph.D. degrees in Computer Science from the Department of Computer Science, University of Bath, UK, in 2003 and 2008 respectively; prior to his doctoral studies, he obtained a Diploma (M.Sc.) degree in Computer Science from the Computer Laboratory, University of Cambridge, UK, in 2004. After his Ph.D., he continued in the same department to become a research and teaching fellow. Since 2011, he became a lecturer (assistant professor) at School of Electronic Engineering and Computer Science, Queen Mary, University of London.



David Pickup David Pickup is currently studying for a PhD within the Computer Science department of the University of Bath supervised by Prof. Phil Willis. He holds an MEng in Computer Science from the University of Bristol and is a member of the Media Technology Research Cente (MTRC), and the Centre for Digital Entertainment (CDE). His research is centered within the fields of computer graphics and computer vision, with special focus on video-based water reconstruction, and using this reconstruction to guide computer graphics animations.

Chuan Li Chuan Li is a research officer in the Department of Computer Science, University of Bath. His research interests include computer graphics, computer vision and machine learning. He published papers for modeling nature scenarios, for example trees and water, using consumer level input devices such a single video camera. He is also interested in the problem of urban modelling.

Paul Rosin Paul Rosin is a Professor at the School of Computer Science, Cardiff University. Previous posts include lecturer at the Department of Information Systems and Computing, Brunel University London, UK, research scientist at the Institute for Remote Sensing Applications, Joint Research Centre, Ispra, Italy, and lecturer at Curtin University of Technology, Perth, Australia. His research interests include the representation, segmentation, and grouping of curves, knowledge-based vision systems, early image representations, low level image processing, machine vision approaches to remote sensing, methods for evaluation of approximation algorithms, etc., medical and biological image analysis, mesh processing, non-photorealistic rendering and the analysis of shape in art and architecture.

Peter Hall Peter Hall is Associate Professor in the Department of Computer Science at the University of Bath. He is also director of the Media Technology Research Centre, also at Bath. He founded to vision, video and graphics network of excellence in the UK, and has served on the executive committee of the British Machine Vision Conference since 2003. He has published extensively in Computer Vision, especially where it interfaces with Computer Graphics. More recently he is developing an interest in robotics.