# Constructing And Rendering Vectorised Photographic Images

J.W. Patterson[*], C.D. Taylor[†], P.J. Willis[‡]

[*]School of Computing Science
University of Glasgow
jwp@dcs.gla.ac.uk
www.gla.ac.uk/schools/computing

[†]School of Mathematical and Computer Sciences
Heriot-Watt University
C.D.Taylor@hw.ac.uk
www: www.hw.ac.uk/macs

[‡]Media Technology Research Centre
Department of Computer Science
University of Bath
P.J.Willis@bath.ac.uk
www.bath.ac.uk/media

## Abstract

We address the problem of representing captured images in the continuous mathematical space more usually associated with certain forms of drawn ('vector') images. Such an image is resolution-independent so can be used as a master for varying resolution-specific formats. We briefly describe the main features of a vectorising codec for photographic images, whose significance is that drawing programs can access images and image components as first-class vector objects. This paper focuses on the problem of rendering from the isochromic contour form of a vectorised image and demonstrates a new fill algorithm which could also be used in drawing generally. The fill method is described in terms of level set diffusion equations for clarity. Finally we show that image warping is both simplified and enhanced in the vector form and that we can demonstrate real histogram equalisation with genuinely rectangular histograms straightforwardly.

**Keywords:** Model-based coding, Rendering, Level Sets.



*First presented at the European Conference on Visual Media Production (CVMP) 2009, extended and revised for JVRB*

## 1 Introduction

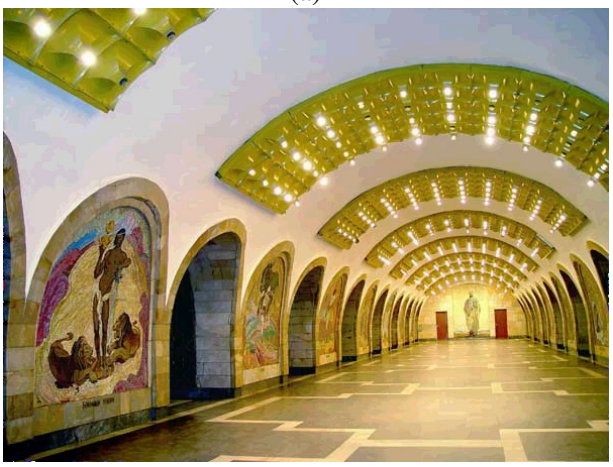
A common problem in distributing digital images and movies is that of catering for varying image or film formats. For example a short sequence of a feature film may be shown on standard TV ($768 \times 576$), HDTV ($1920 \times 1024$), internet video (various), or even mobile phones (anything from $384 \times 256$ upwards). If shown for publicity reasons the producers will want this to be shown at the best quality possible. In effects houses which concentrate on advertising a significant proportion of time is spent just converting between the various digital formats on which the advertisement is to be shown. The problem arises because all digital

images have to be sampled in order to be seen at all, but different kinds of display device have, of necessity, to show the images at differing resolutions.

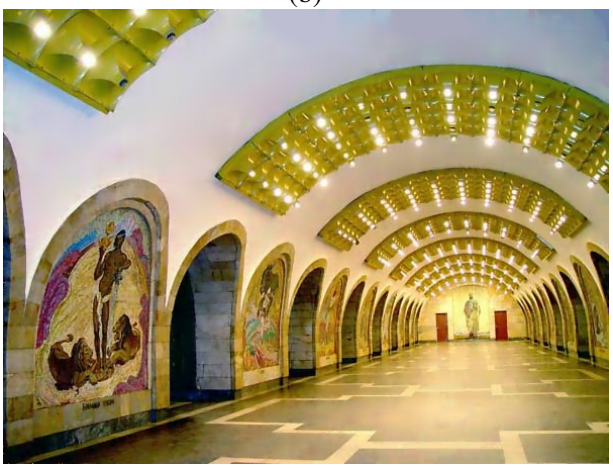
*Vector* formats, historically associated with drawn images rather than with photographs, can provide a resolution-independent format but none of the existing fully automatic fill rules for vector formats work well on photographic images. This paper describes a new fill regime based on diffusion which results in images which are visually indistinguishable from their originals after conversion into and out of a vector representation. There are a number of systems and plug-ins available to turn images into vector form (e.g. Adobe Live Trace$^{TM}$) but they are all compromised by the absence of a good rule for determining varying colour values in a continuous field. Tools like Live Trace$^{TM}$extract isochromic contours from sampled images but have to extract a sufficiently large number of contours to preserve the illusion of a smooth surface on a sampled display. This is because the usual rule for automatically filling between contours is to provide a constant colour (here called *flat fill*) in the viewable rasterised version of the image so the results can be thought of as providing a series of step changes in colour values rather than a continuous variation in those values. In Figure 1(b) we have used a flat fill regime on an image which was vectorised for a diffusion-based fill regime of the kind described later in this paper. The diffusion-based approach allows for the use of fewer contours without compromise to the appearance of smoother parts of the image and this has shown up a particular weakness of flat fill in the form of visible bands of colour changes (Mach bands). Indeed the usual application of such systems is to produce an artistic effect rather than a realistic outcome.

In this paper we show how to resolve this problem. As the contours can be thought of as a model of the image they need to be converted to a sampled image format in order to be seen on raster display devices. Thus a rendering process, entirely analogous to rendering processes used in 3D graphics, is required. The problem here is, if flat-fill is insufficient, how do we determine the intermediate pixel values between contours when rendering back to the sampled form? This problem and this paper's contribution to its solution are best illustrated in Figures 1(a)-(c) above. In Figure 1(b) the straightforward approach of flat fill has been taken. More explicitly each contour footprint, here defined as the image region uniquely enclosed by a contour, has been filled with a constant 'aver-

(a)



(b)



(c)

Figure 1: Top to bottom: (a) Original digital photo (b) Rendered: flat-fill (c) Rendered: interpolated fill

age' colour. In Figure 1(c) a diffusion-based interpolation technique (described in section 4) has been used instead. Each image was rendered from the same contour set. In Figure 1(b) the different outcomes are quite noticeable; flat fill shows Mach-banding artefacts which are wholly absent with the interpolated fill of Figure 1(c). As can be seen in Figure 1(c) the interpolating fill technique produces satisfactory results even though a quite simple rule has been used to determine which levels to use in the vector form.

A major problem in determining this vector form is that of determining how many contours to use. A naïve approach would be to use one contour for every unit of colour quantisation but this not only produces far more contours than are needed for a visually faithful reproduction of the image but also far more detail in defining each contour than is needed. If flat fill is used within contour boundaries the inefficient naïve approach is the only technique which will assure an artefact-free result. Techniques based on morphological principles (e.g. [Ser82]) do this and, while we reference these techniques as part of the historical record, we would like to draw a distinction between the morphological approach and ours, notably in terms of the number of levels required for a visually satisfactory rendering. In such an approach all that would be needed for a visually faithful rendering would be to fill the contour with an 'average' colour quantised to be the border colour. In fact far fewer contours are needed in practice as we will show here.

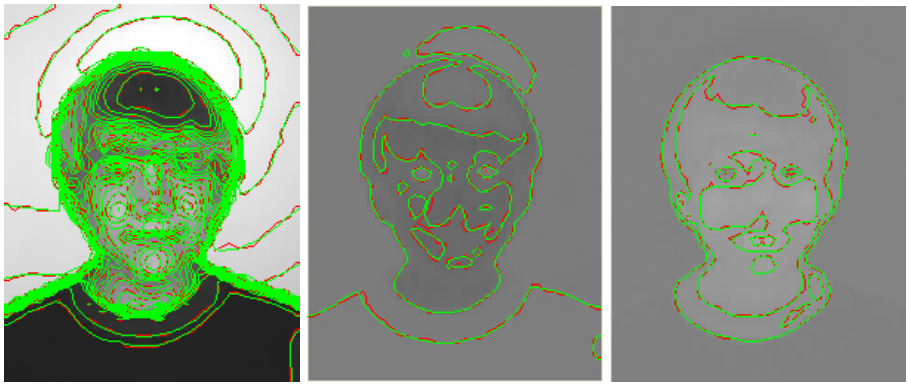
To illustrate the contours without too much visual confusion we chose the out-of-focus image Figure 2(a). Here the images have been rendered from contour sets representing the YUV components of a colour image. As can be seen in Figure 2(c),(d) and Figure 2(f),(g) there is very little detail in the U, V components so the size of the dataset is dominated by the Y-component. (This remains true even with a sharp, high-definition image.) The Y component (grey-scale) is held in levels-of-10 256 level quantisations, i.e. the colours 127, 127+10 etc. 127-10 etc. of which there are 25 altogether; while the U,V components, being more subtle in variation, are held in levels-of-5 (so 127, 127-5, 127-10 etc.) of which there are 49 altogether. This selection of contour levels has proved satisfactory for most of the images we have used here. We use Bézier chains to define contour borders, initially ignoring continuity between adjacent segments. The contours for the YUV components of Figure 2(a) are shown in Figure 2(e)(f)(g). The red contours are unsmoothed versions of the green ones. Smoothing is achieved by 'snapping' together endpoint tangents if they fall within a certain threshold of being directionally opposite. This threshold value is one of several coefficients which can be modified to improve the image in one way or another.

This paper is structured as follows. The next section 2 covers previous work in this area, some of which reaches back to the earliest days of computer graphics. Section 3 discusses the main features of the vectorising codec we have used here, which has an encoder which works from a raster image to produce a vector image format and a decoder which renders the vector format back into a raster image. Section 4 discusses the decoding stage in more detail, and in particular the principles behind the diffusion-based interpolating fill process. We will see that this fill could be used just like any other area fill technique used in rendering vector format images, and on its own would give a drawing program (or Scalable Vector Graphics SVG[DHH02] - interpreter) the ability to reconstruct contourised images to whatever degree of fidelity is required. Section 5 discusses some common image manipulation functions, including histogram equalisation, and here shows the consequences of producing truly rectangular histograms which are unachievable using sample-based techniques. Section 6 reviews where this work places vector image formats and section **??** will conclude the paper.

## 2 Previous work

Vectorising (contourising) as applied to an array of sample points is a technique whose origins go back to geographical information systems [WW67], where contour maps were to be produced from spot height surveys, and was first applied directly to photographic images by Matheron[Mat75] and later by Serra[Ser82] (these are the early references to the morphological approach). At about the same time Nakajima et al.[NAT83] (a more accessible reference is Agui et al.[ASNA87]) proposed an approach more closely allied to the techniques of computer graphics. The fill method used in image reconstruction in all these references is flat fill. More recently Price and Barrett[PB06] and Sun et al.[SLWS07] have proposed methods for generalising from flat fill while keeping the same number of contours[SLWS07] or sample points[PB06] by building an adaptively subdivided mesh where colours are associated with mesh inter-

(a) Colour image (b) Y-component (c) U-component in greyscale (d) V-component in greyscale



(e) Contourised Y for 2(b) (f) Contourised U for 2(c) (g) Contourised V for 2(d)

Figure 2: Colour components and their contours

section points, although some interaction is required to determine the starting shape of the mesh. Both the method of contour finding and mesh generation (Live Trace™[PB06, SLWS07] and gradient mesh tool[SLWS07] respectively) are available in commercial drawing packages but the papers focus on smooth colour interpolation and mesh optimisation for minimal dataset size. However this approach simply swaps one set of sample points for another more feature-oriented set, as befits the type of calculation they want to do, and offers no help to calculations like histogram equalisation or processes involving preserving features of the isosurface topology (The isosurface is defined by a given set of isochromic contours selected from the set of quantised colours used).

Another approach which does not use contours, but rather chooses edges as the key feature, is that due to Orzan et al.[OBBT07, OBW+08]. Here the idea is to use edge-lines as the vectors and to decorate the lines with colour data. This is then propagated away from the edge using a Poisson equation. When used for images the edge lines correspond to discontinuities in otherwise smooth shading and reconstructed images look a lot like their original forms although the unconstrained use of Poisson equation diffusion results in quite inaccurate diffusion boundaries. It is the insight of Lindeberg [Lin98] (and others) that this inaccuracy tends not to be noticed which Orzan et al. are exploiting here. They also note[OBW+08] that a similar decorated-edge representation can be used to produce smooth-shaded images of a kind difficult to generate by other means. Similarly the diffusion method described in this paper could be used to generate different forms of smooth-shaded synthetic vector image although the method of control would be quite different.

In the end any accurate method of vectorising a photographic image needs to have some kind of interpretation of just what a pixel is and in essence we make a different interpretation of an image pixel on input to the interpretation made on output. Two papers which discuss this problem in ways we pay particular attention to are those due to Blinn[Bli05] and to Smith[Smi95]. Blinn discusses eleven distinct ways in

which to consider what a pixel actually is and this includes a discussion of the relationship between a sample and a sensor which generate samples, while Smith makes a strong argument for not considering a pixel as being a square over which some simple form of integration is done (e.g. a tent filter). Pixel generation (in our decoder) uses supersamples under the footprint of a convolution kernel, which is quite different to the assumptions about input although the nature of the input environment may be taken into consideration when deciding what the pixel value might be, for example to use noise statistics to determine how closely to approximate the round off in quantisation. In fact we have combined regular 4 x 4 supersamples (sometimes 9 x 9 supersamples on a 4 x 4 sampling grid) by integrating over a square in all our images in this paper without being caught out, but more stringent sampling or averaging regimes are not excluded by our approach.

# 3 Main features of our vectorising codec

An image vectorising codec starts from a sampled image, typically one obtained from a digital camera, encodes it into an annotated contour set (derived as and collected into level sets) and subsequently decodes it back into a sampled image after whatever image transformation processes required are applied to it. A diagram of the main encoder stages used to produce the images in this paper is shown in Figure 3, and the corresponding decoder stages are shown in Figure 7.

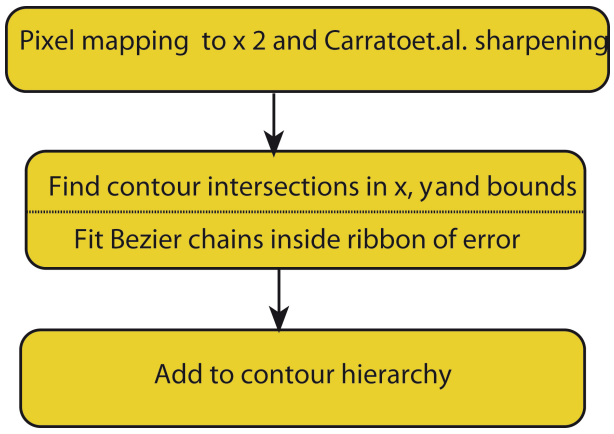



Figure 3: Encoder structure

## 3.1 Contour representation

We will now set out the principal features of our approach. In our codec individual contours are always represented by closed Bézier chains; contours clipped by the image border are completed by the shape of the border segment which falls within the contour footprint. This has certain implications for constructing the intermediate vector format and for the diffusion process involved in decoding (section 4.2).

Contours are found by first finding where they intersect lines between sample points derived from the original pixels. A key aspect of the input process is the explicit assumption that the pixel is contaminated by noise which can arise from any source, quantisation, sensor noise, even numerical inaccuracy, so is essentially of unknown origin. For example when considering the degree of accuracy to which the isosurface is modelled the strictest requirement we can safely make is that the isosurface model lies everywhere inside the error bounds of the pixels.

Pixel error can be modelled in a number of ways, essentially either globally or locally. The accuracy of the value derived from the model is not critical although too crude a model could result in retaining image noise in the final result or a result which loses detail. While more accurate local approaches are covered in Patterson and Willis[PW06] we should say that all the images in this paper were generated assuming a simple global noise value ($\pm$constant around each pixel value) without apparent loss of detail due to that assumption. If we are to attempt to preserve noise statistics in the final image, as suggested earlier, more accurate, local, methods will be needed.

Error terms $\epsilon$, however derived, can be converted into spatial errors $\delta x$, $\delta y$, in the $x$ or $y$ directions by applying the formulae:

$$\delta x = \epsilon \left( \frac{\partial \phi}{\partial x} \right)^{-1}, \delta y = \epsilon \left( \frac{\partial \phi}{\partial y} \right)^{-1} \qquad (1)$$

Here $\phi = \phi_{x,y}$ is the piecewise continuous approximation of the 'true' isosurface. We now need to introduce some notation, as shown in Figure 4.

## 3.2 Pixel extension (pixel mapping)

The first stage in the encoder is described as 'pixel mapping'. Here we double the image samples in each direction so that each set of 2 x 2 samples provided to the contour finder corresponds to a 'map' of the original pixel. However the new samples are all calculated

by a non-linear process due to Carrato et al.[CRM96] which emphasises edge detail, as more conventional interpolation (we use linear interpolation elsewhere for reasons which will be discussed shortly) does not introduce any new information.

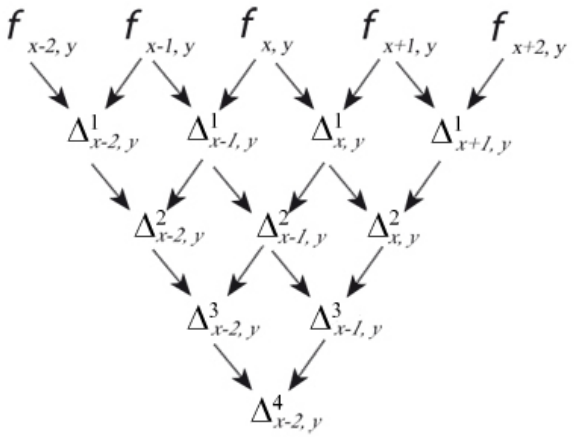



Figure 4: Forward differences table in $x$

The re-sizing step is in part intended to allow some distance between contours we take as adjacent although inevitably reconstructed edge values are often not identical with their original values (this is not noticeable to the naked eye) and the Carrato-sharpening is intended to defeat an ambiguity problem which arises in interpreting contour trajectories with the "marching squares" algorithm because of uncertainty over the centre sample in a pixel map. Carrato will bias the solution one way or another depending on surface topology and, although it would seem as though we have merely pushed the problem down to the next (second) level in fact linear interpolation is quite adequate for the second order case. (Surfaces tend to flatten out locally with more interpolants.) The scheme is illustrated in Figure 5(a)-(c).

Here Figure 5(a) shows what we mean by modelling an image in terms of pixels, which may have many interpretations[Bli05], but commonly (despite Smith[Smi95]) as little squares. For our purposes a more appropriate model would be point samples addressed from the pixel centre (Figure 5(b)). Figure 5(c) shows one of the interpolated pixel values in red. Using the Carrato formula this value would be calculated using the following indices.

$$\phi_{x+\frac{1}{2},y} = \phi_{x,y} + \frac{k_1+1}{k_2+2}.\Delta^1_{x,y}$$

where

$$k_1 = p.\left(\Delta^1_{x-1,y}\right)^2$$

and

$$k_2 = p.\left(\left(\Delta^1_{x-1,y}\right)^2 + \left(\Delta^1_{x+1,y}\right)^2\right)$$

For the images in this paper we used the value $p = 0.5$. Similar equations prevail in the y-direction. The errors in the interpolated values are determined by subtracting the maximum and minimum interpolant values, using the limiting errors in the pixel values contributing to the result. For these differences, the index map for the error calculation is as follows.

$$\Delta^1_{x,y} \pm (|\epsilon_{x,y}| + |\epsilon_{x+1,y}|)$$

where the error $\epsilon_{x,y}$ is the (designated) error in $\Phi_{x,y}$. The resultant error is then associated with the interpolated pixel which participates in all subsequent calculations indistinguishably from the originals.

The spatial errors in equation 1 are quoted in terms of continuous derivatives but once again we use first differences to generate the needed values, this time central differences, i.e. we use the formulae

$$\delta \approx \epsilon\, left(\Delta^{lC}_{x,y}$$

where

$$\Delta^{lC}_{x,y} = \frac{\Delta^1_{x-1,y} + \Delta^1_{x,y}}{2}$$

(and corresponding formulae for the y direction). In the following discussion we will take $\epsilon = \pm 0.5$, the quantisation error, which is the smallest value of $\epsilon$ we can assume, but when encoding the image examples we have used here we essentially scaled $\epsilon$ by a constant amount chosen subjectively and intended to reflect our sense of how noisy the image was. Most digital images available to us have been through a process of JPEG encoding. It would be possible to trace through the decoding process to determine the per-pixel quantisation error which is typically image-dependent and spatially variable in that image, although we did not actually do this. For us the correct way of handling the effects of JPEG encoding is simply to use these quantisation errors as a lower bound for $\epsilon$ if other methods of estimating it are in play.

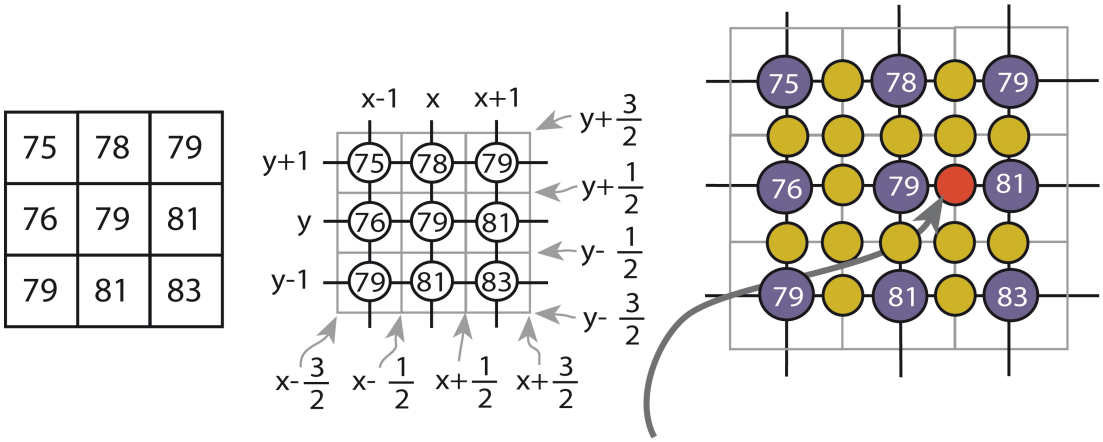



Figure 5: Image patch as (a) Pixels (b) Addressed point samples (c) Pixel maps

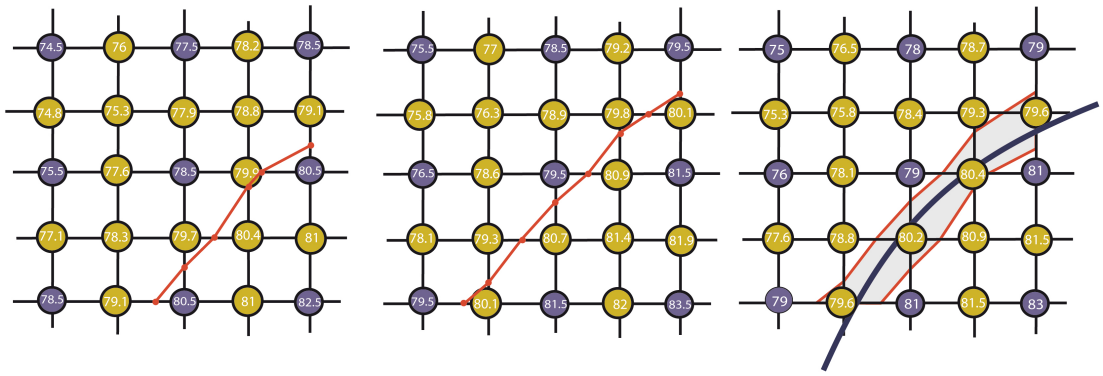


Figure 6: Pixel tile with (a) $\epsilon = -0.5$ (b) $\epsilon = +0.5$ (c) $\epsilon = 0.0$

### 3.3 Constructing contours

To show how a contour is determined we will use an example in which the error term is taken to be the quantisation error ($\pm 0.5$). This is shown in Figure 6. If we determine the path of a contour in terms of its intersections with the borders of a box whose corners are decorated with sample values we can interpolate between sample values to get an 'exact' solution, thus allowing the 'exact' computation of the lower bound of the contour trajectory, Figure 6(a), and the upper bound, Figure 6(b). When these are transferred to the pixel tile with no error bound ($\epsilon = 0$) they delimit the region we refer to as the ribbon of error within which any contour trajectory is equally valid. This is true for any realistic calculation for the pixel error. In fact we associate these errors with points on the contour line when fitting the Bézier chain but any fitting algorithm (e.g. the method outlined by Schneider[Sch90] or by Vansichem et al.[VWR01] has to take into account different values for that error around each sample point. One way to relax this condition is to multiply the derived error values with a constant and the consequence will be to reduce the number of segments in the chain and increase its smoothness.

It turns out that failing to take into account the presence of noise results in large numbers of Bézier segments in every contour as it twists and turns around single pixel-sized 'features' which are no more than noise-induced deviations from local correlation.

If instead we account for noise adequately we get much smoother curves (with many fewer segments) whose smoothness has, up to a point, no perceptible effect on the resultant render. As indicated earlier the interpolation formula used is that for linear interpolation, that is by solving in the appropriate direction for the point where the linear interpolant along x or y is equal to the sought contour value, and then interpolating between the sample errors by the same amount. This is entirely equivalent to a process of solving for two isosurfaces, and is in fact safer, numerically. If

we had used a local error measure then we could have determined which degree of interpolation was appropriate on a solution-by solution basis by finding which difference approximated to zero within the calculated error bounds for that difference[PW06]. However past experiments with a pixel-level noise estimator have showed that for the majority of cases (approximately two-thirds in typical images) only linear interpolation could be justified and the rather basic assumptions about noise made here would not justify higher order interpolation anywhere. The resulting polyline approximation to the 'true' contour is simplified by finding the Bézier chain with the fewest segments which fit within the error ribbon the polyline approximation defines.

In an encoder the contour values can be determined by a blind strategy or an adaptive strategy. In this paper we have used a simple but (reasonably) effective blind strategy of pre-selecting the values to be found. As a consequence all the images in this paper are represented by the same choice of contour levels and all the contours are found at the same time by a single scan of the image from top to bottom. Here we examine each pixel to see which contours pass through a bounding box around its centre and then join up the contours by matching adjacent bounding box edges. This scan-based approach is only really possible with a blind strategy as an adaptive strategy will of necessity contain a stopping condition based on testing the need for the contour loop under consideration.

## 4 Rendering between nested contours

The outcome of the process is a hierarchy of contours (including contours R and S) defined in terms of the relation R encloses S defined as follows in terms of closed connected regions such as A . To define $encloses(,)$ we start with the well-known inside test $inside()$ where $p\ inside\ A \supset W_A(p) = s$ where $W_A(p)$ is the winding number for p in the closed region A and s is consistently +1 or -1 for every point p in the region. Now we need $footprint()$ defined as:

$$footprint(A) = \{p | p\ inside\ A\}$$

and hence:

$$A\ encloses\ B \supset \left\{ \begin{array}{l} B \subset\ footprint(A) \wedge \\ \exists C, A\ encloses\ C \wedge C\ encloses\ B \end{array} \right\}$$

So $A\ encloses\ B$ is an ordering relation between individual closed contours in each level and determines all the contours B which are enclosed by A and by no other contour. Rendering the vector format consists of rules for drawing and filling these contours in order. The decoder, shown here as Figure 7, 'simply' applies the fill rule for the footprint of each pair of contours in the hierarchy as defined by the footprint of the enclosing contour subtracted from the footprint of all the contours it encloses directly. So although (here) contours are generated in levels (i.e. all the contours at a given level) they are ordered in terms of individual pairs of contours.

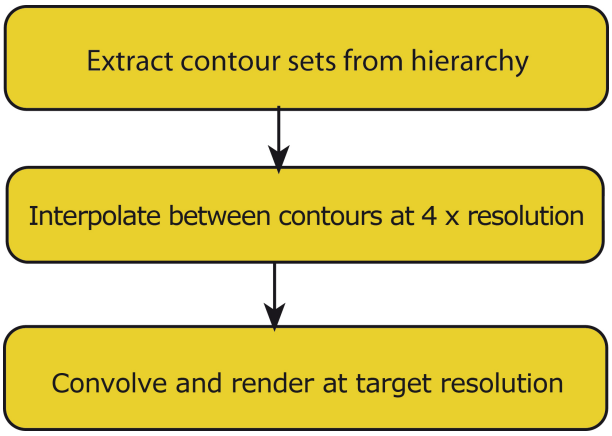
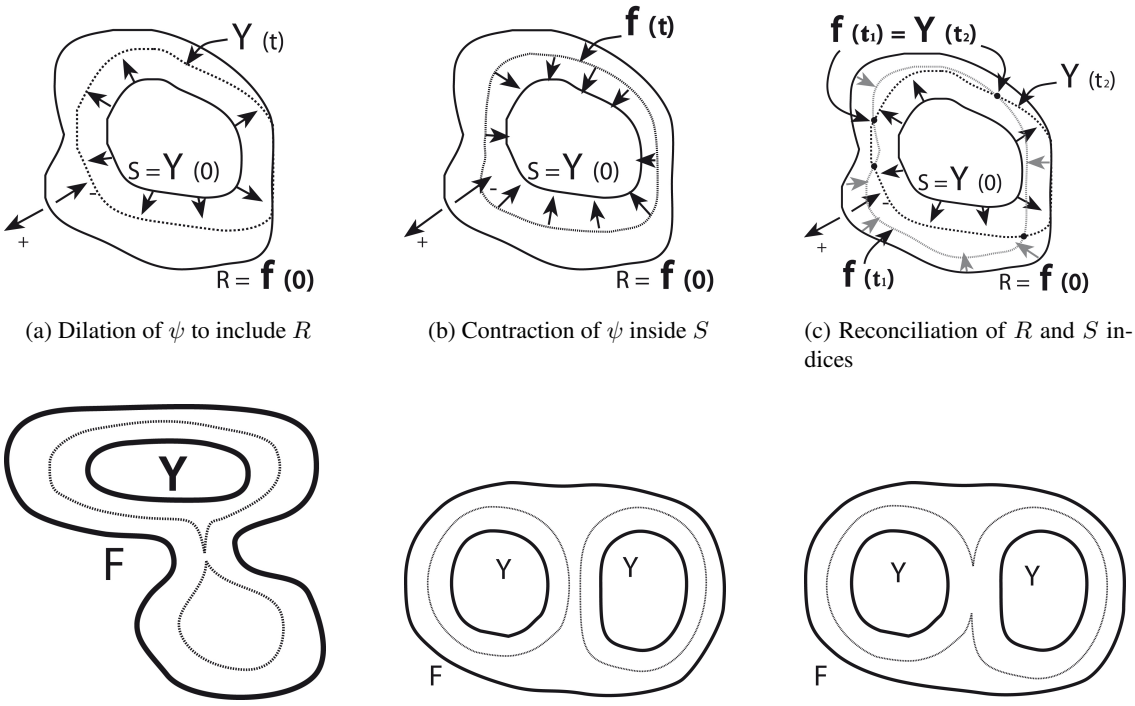



Figure 7: Decoder stages

The principal issue in rendering is to use a process which mimics to some degree the fall-off in values of pixels from a higher level to an adjacent lower one (or vice-versa). The intention is to develop a simple diffusion-based fill algorithm between levels, as defined by level lines (isochromic contours). We will first develop the idea in Level Set terms and then show how to implement it without having to solve the differential equations which the invocation of level sets implies. The reason for doing this is that Level Set theory makes the issues clear in a direct and easily visualisable manner but the complexities of solving the equations led us to use known fast, scanline based methods to give us quick renders.

We derive the formulae in terms of a simple case (Figure 8) of a single outer contour with level value $R$ surrounding a single inner contour level $S$, and then generalise. We want to arrange for the inner contour $S = \psi(0)$ to first expand (in the terminology of Vincent[Vin93]: 'to dilate') at a uniform (unit) speed until it wholly contains $R$ (Figure 8(a)). At all times




(a) Dilation of $\psi$ to include $R$

(b) Contraction of $\psi$ inside $S$

(c) Reconciliation of $R$ and $S$ indices

(d) Pinching of isochromic contour by shape

(e) Effect of two interior contours on intermediates

(f) Effect of two interior contours on intermediates

Figure 8: Contour creation

the level line for $\psi(t)$ over the interval $0 \leq t \leq 1$ gives the shape of the intermediate dilation at instant $t$ which we note is wholly dependent on the shape of $S$ and has no connection with the shape of $R$. If, at the same time, the corresponding level line for $R$ is contracted ('eroded'[Vin93], as in Figure 8(b) at uniform speed until it falls wholly within $S$ then the level lines for intermediate erosion at time point $t_2$ , say, will intersect the level lines for dilation in a range around another time point $t_1$. These time points $t_1$ and $t_2$ now correspond to the times taken to reach the nearest points on R and S respectively, so define linear distances within $R - S$ ($R$ with $S$ removed from within it) which can be used to interpolate colours associated with $R$ and $S$ respectively at the points of intersection of these two curves. In fact we can go further and say that if we take any point $p$ inside $R - S$ then the time taken to reach it from $R$ is the morphological distance from the boundary of $R$, and the time taken to reach it from $S$ is the corresponding distance from $S$. So by using these distances as interpolants we get (within later caveats) the right intermediate colour for the point $p$.

### 4.1 Level Set diffusion formulae (expansion)

Reviewing and extending what we have said so far, for any point inside the region $R - S$ the shortest distances to the boundaries of $R$ and $S$ respectively determine the interpolation of the colours associated with the bounding levels $R, S$ at that point. These interior colours will usually vary linearly from the values associated with one contour to the other, but this only happens if the geodesics running through the points in question are straight lines, i.e. the boundaries are not occluded from the point. In more complicated situations this approach generates interior colours which vary as though affected by surface tension, which is likely to fit what is actually found. This is shown further in Figure 8(d) where the intermediate contour or isochromic line is actually split into two and there is a local 'bubble' in the middle of the lower lobe of the outer contour. An analogous situation is shown in Figure 8(e) and (f), where there are two interior contours in the footprint of the outer contour. In Figure 8(e) the isochromic line has not yet joined up while it has done so in (f).

Taking the outwards direction as positive (as shown in Fig 9(a), the level set equation for the expansion

(dilation) of $\psi$ is:

$$\frac{\partial \psi}{\partial t} = K \frac{\nabla \psi}{|\nabla \psi|}$$

where

$$K = \begin{cases} 1 & \text{if } |dist_{R-S}(\psi(t), R)| > 0 \\ x & \text{otherwise} \end{cases} \quad (2)$$

Here $\psi(t)$ is the expansion of $\psi(0)$ clipped by $R = \phi(0)$.

The function $dist()$ gives geodesic distances of points in R, R(a) say, from S using the value of t at the points at which $\psi(t) = R(a)$. It is defined formally in the Appendix but may informally be thought of as the geodesic distance between two points, or a point and a boundary within an irregular, complete enclosure on a finite plane.

### 4.2 Contraction formulae and reconciliation of outcomes

We can obtain the morphological distance fields for $-R$ and $+S$ by evaluating equation 2 and its matching partner for the erosion[Vin93] of $R$, equation 3 as in Figure 3(b):

$$\frac{\partial \phi}{\partial t} = -K \frac{\nabla \phi}{|\nabla \phi|}$$

where

$$K = \begin{cases} 1 & \text{if } |dist_{R-S}(\psi(t), S)| > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

This will give two Euclidean distances $t_1$,$t_2$ , where $\psi(t_1) = \psi(t_2)$ at zero or more points inside $R - S$ (as in Figure 3(c)) so we can calculate the colour values $C$ of those points from the colours

$C_R, C_S$ associated with level lines $R$ and $S$ respectively as:

$$C = C_R \left( \frac{t_2}{t_1 + t_2} \right) + C_S \left( \frac{t_1}{t_1 + t_2} \right)$$

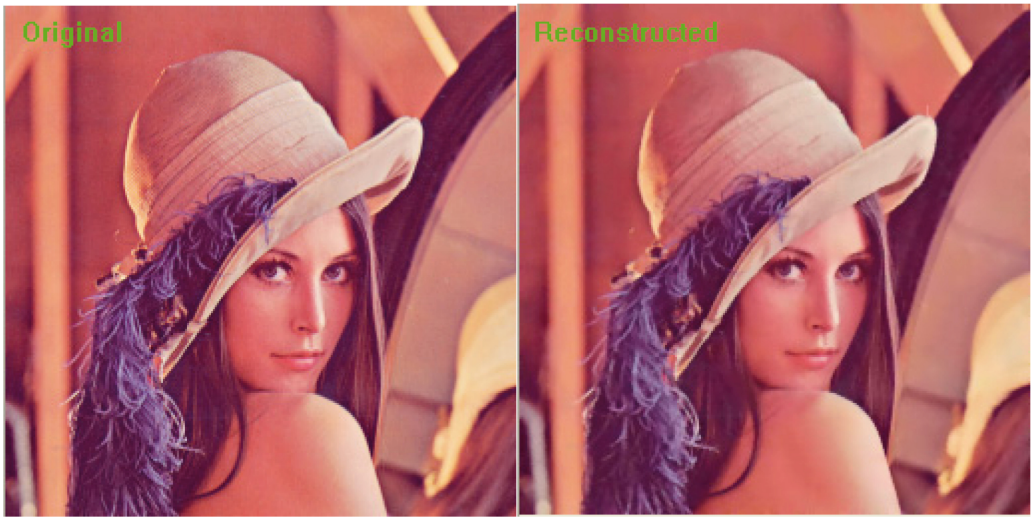
We have used diffusion twice to give us morphological distances from each point in space in terms of the time to reach each of the two levels (here the speed of travel is unity so time = distance). When normalised by the sum of the distances this gives us an interpolation ratio between the two contour values which is linear for simple geometries but quadratic with a positive curvature - quite similar to the effects of surface tension - when the contour geometry becomes complicated. We refer to this process as double diffusion and note that isochromic lines are in effect interpolants between the shapes of the inner and outer contours, so it should properly be called *double diffusion interpolation*.

For this paper we used a computationally simpler measure than Euclidean distance, namely *Manhattan distance*, calculated outwards (inwards) from a border defined in terms of those pixels which contained the border contour. The Manhattan distance can be calculated like a fill process in which successive erosions or dilations define an ascending index starting at 1. Although the Manhattan distance is always an overestimate this tends to get normalised by the division of indices calculated in the same way. If the calculation of dilation (or erosion) is carried out in a quantised manner this naturally supports Manhattan distances, but if it is carried out continuously (e.g. by equations 2 and 3) this naturally supports Euclidean distances and the precise calculation of interpolants which are smooth everywhere.

## 5 Example applications

Apart from Figure 1 all the examples showing our interpolation approach are applied to the reconstruction of $YUV$ images with $Y$ at intervals of 10 and $UV$ at intervals of 5. In Figure 9(a) we show an original pixel image (the standard test image 'Lena') and its re-rendered equivalent in Figure 9(b). At intervals of 10 between $Y$ levels the re-render is visually indistinguishable from the original (input spatial resolution $256 \times 257$). Additionally we have shown in Figure 9(c) and (e) two portions of 'Lena' scaled by 4 and 8 respectively using bilinear interpolation (which is usually preferred in the film industry over higher order methods, because it gives a more 'punchy' image). As one expects the detail becomes more blurred. By comparison we have scaled the control points for the contours in Figure 9(b), again by x 4 and x 8 and rendered these as in Figures 9(d) and (f) where we can see that more detail has been carried into Figure 9(f) than in Figure 9(d).

For our second application we chose a different kind of operation, histogram equalisation. The purpose behind histogram equalisation is to adjust pixel values so that each sub-region of the image yields equal energy. Properly histogram-equalised images should show the highest contrast everywhere in the image and this is




(a) Original 'Lena' image (b) Rendered from contours (x1)

(c) Original x4 (bilinear) (d) Contour control points x4

(e) Original x8 (bilinear) (f) Contour control points x8

Figure 9: Rendering 'Lena' image with various techniques

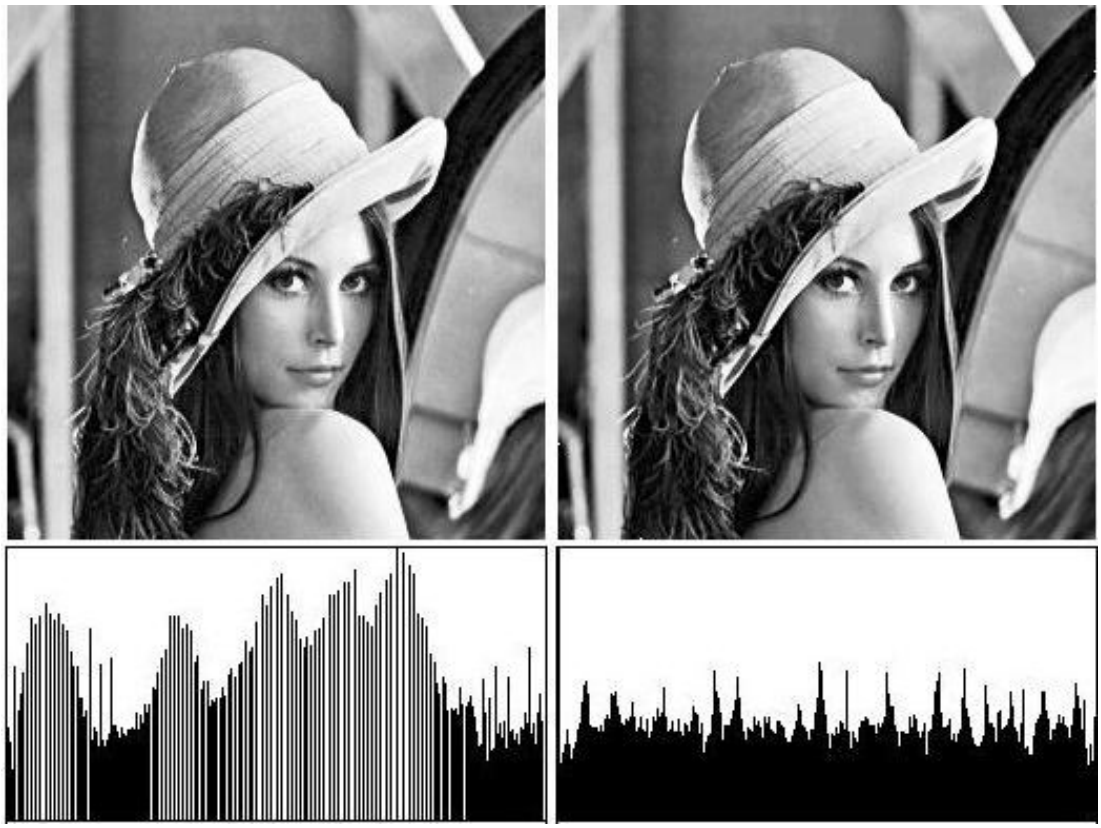


Figure 10: (a) Histogram equalised by pixel (b) Equalised by level reassignment

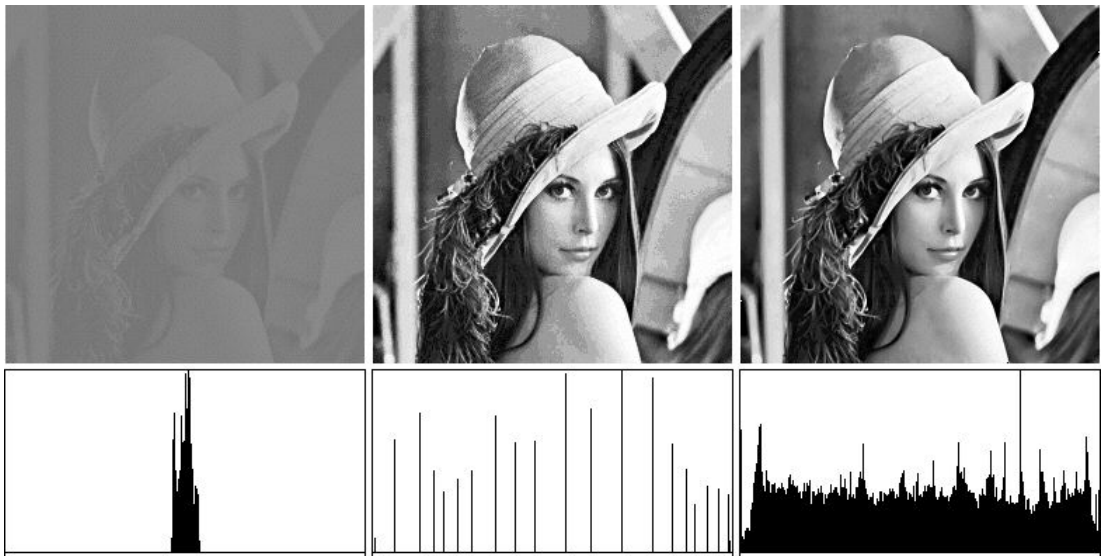


Figure 11: Histogram equalisation from reduced image contrast

supported by a rectangular histogram across the range, particularly for the Y component of a YUV image. Unfortunately pixel-based histogram equalisation usually only manages to achieve the sort of result in Figure 10(a). The diffusion interpolating technique however does not require level lines to be associated with integral values although one might start out that way. Instead one could determine what level values would give the nearest to a rectangular histogram. The neatest way of doing this would be to start off with contours of values which vary in powers of 2 around 127, i.e. 127, 63, 191, 31, 47, 159, 223 etc. The 127 contour should partition the image area exactly in half and if not, its value needs to be reassigned to whichever contour comes closest to achieving that partition, if necessary generating contours (levels) to ensure that partitioning proximity.

We re-assign the levels by indexing the inverse function $H^{-1}()$ by the proportion p of pixels actually covered ($0 \leq p \leq 1$). Here $H(l) = 2^{ln(l+1)-8}$ so $H^{-1}(p) = p*2^8-1$ (for an 8-bit per primary quantised colour space). Figure 10 shows the results of performing histogram equalisation in this way. Here the first row shows the resultant images and the second row shows the corresponding histograms. It is clear that the right hand image Figure 10(b) has the strongest contrast enhancement of the two images and in particular lacks the artefacts of the left image Figure 10(a) which are in the main caused by the gaps introduced by pixel reassignment in the histogram.

## 6 Discussion

Vector formats for photographic images have been studied for various purposes since the mid 1970s and there are broadly two approaches, the morphological approach which in effect requires every individual quantisation of the colour level to be represented, and the topological approach which attempts to model the isosurface as economically as possible. Level sets, as we have used them here, are a bridge between the two approaches and can benefit from results in either. Many image manipulation operations on this form are both simpler and seem to give better results (as here, warping and histogram equalisation) than their raster equivalents. One kind of transformation is particularly straightforward, that of varying the colour depth resolution in the resulting image. This is because the final samples for the observed image are calculated as a convolution of samples into the continuous field and these need then to be quantised to whatever colour depth is required. Thus the vector form is independent both of spatial and colour depth resolutions in the original input.

However there are residual problems with the vector approach which can be summarised in terms of conversion speed and file size. Conversion times in and out of the vector format are approximately linear with input image size although it is known that images with a lot of high frequency detail take longer to encode and decode than images with a more usual distribution of frequencies. For example the 'mandrill' image (another one of the standard test set) takes twice as long to encode as 'Lena'. On a 500MHz PC 'Lena' at $256 \times 257$ took 15 seconds to encode and 20 seconds to decode, but this is without any graphics acceleration assist. The codec is (by intention) well-suited to streaming and parallelisation. Our view is that, given the degree of support available for graphics processes, these times will be significantly improved in practice. On the other hand the fixed contour level setting strategy resulted in file sizes 10x larger than their pixel equivalents which we did not attempt to address in the work being reported here. However, the results of Lindeberg[Lin98] suggest we have been far too conservative in fixing the local resolution of contour segments in smooth areas. If we compute a resolution measure which scales with local smoothness we should be able to significantly reduce the number of segments per contour. We have also (knowingly) been far too conservative in the numbers of contours we find, possibly by as much as a factor of 20, but it will take a (much more complex) adaptive encoder to find the local optima. Compression is an obvious focus for future work.

Our original concern was to be able to reproduce photographic images from contour form so that they looked visually indistinguishable from the original. In this we were generally successful. A particularly demanding example is shown in Figure 12 where a photograph of a seagull, Figure 12(a) has been vectorised and decoded again using the exact regime described in this paper. While the results impressed us they nevertheless show that the method of setting fixed contour levels to find is not perfect (the insets of the same regions in Figure 12(b) and 13(d) which are double image size differ visibly).

Reliably better results could be obtained with an adaptive encoder which starts as we have done here by encoding contour 127 then splitting the intervals on ei-

Figure 12: (top to bottom): (a) Seagull (original image) with neck feather detail from below beak (b) Vectorised image render with same feather region

ther side etc. Such an adaptive encoder (e.g. [PW06]) would maintain a test render and only split an interval further if the pre-render resulted in pixel values falling outside the assumed error bounds around the original pixels. Where this condition happened locally, a local decision to subdivide further could be taken. We estimate that such a codec would take twice as long to encode as our fixed-level encoder. It would take considerably less time to decode, although this time would be more image-dependent than before. File size should be significantly improved also. The gains here also depend on the noise-estimation method used as well as the extent to which the diffusion process mimics the expected variation of pixels values within a contour footprint. Here there are a number of possible improvements to be made.

The main improvement is to use a standard level set approach [Set99] of adding or subtracting (depending on the sign convention used) the traversal speed of the level line with a (usually) small amount, calculated as (say) 0.01 x curvature, where curvature is calculated as $\nabla \frac{\nabla \psi}{|\nabla \psi|}$ This has the effect of smoothing out 'shocks' or discontinuities in the evolving line, which is a common problem in interpolating systems [Ree81]. (We note also that loops are another problem with 2D interpolators but the rules for interpreting Level Set solutions explicitly precludes these under the 'weakest solution' rule; instead the loop is cut off at a point which often leaves a visible shock.) Shocks also arise when two advancing fronts intersect one another, as in Figure 8(e) and (f) but again the 'weakest solution' [Set99] applies to determine a single front. Again the foregoing modification smoothes away the discontinuities, but the calculated indices are no longer computed from wholly linear (Euclidean) distance values. This would only normally be done away from edges so requires a separate edge detection process to work properly. It is also possible to manipulate these indices further so that tangents in the trajectories of index values are matched across contour boundaries to achieve $G^1$ continuity ($C^1$ can be achieved with greater difficulty, typically as a post-process if needed after establishing $G^1$). While we would expect an improvement in image quality (and a matching improvement in file size) by these measures, such improvements are usually visually unnoticeable in an unwarped image.

Our motivation for this work has been based on the intuition that contours will commonly follow the features of objects in the image. We hope in the future to be able to show that 'difficult' operations like matte-pulling and hole-filling will be enhanced by vector formats in addition to the processes whose enhancement we have already demonstrated. Image re-sizing is such an example where the vector format can be exploited to define localised warps aimed at preserving the slope angle at edges. This has the effect of retaining feature sharpness but vector image resizing under various regimes is potentially the subject of an entire paper in itself, so, despite its importance to some industries, this has not been discussed here.

What we have shown already is that there is a viable continuous image format and that it can be used for some conventional operations which are not handled easily in sampled formats. Moreover, while the representation cannot reveal more detail than in the original sampled image, it does offer a robust model of that image, with all the advantages of being able to render it at different qualities for different devices. It thus has the advantages that SVG offers for graphical pictures but with the ability to deliver the full quality of photographically-captured images.

# 7 Conclusions

This paper describes a vectorised image codec which finds contours for non-consecutive quantisations and fills them using a simple-seeming diffusion process during rendering. This two-part process turns out to be surprisingly powerful, allowing full control of the continuity of the rendered isosurface, for example not attempting to enforce more than $C^0$ continuity at edges, although we have not demonstrated this in examples here. We are aware that there is considerable scope for improvement of the codec although the results we obtained here demonstrate or the first time near-perfect reproductions of most photographic images from contour maps. Given the multiple potentials of the approach and given the fact that the 'traditional' objections to vector artwork seem to be melting away we argue that this whole approach to image-making, which banish pixels from anything beyond external representations, deserves closer scrutiny and further work.

# Acknowledgments

We acknowledge funding from the Scottish Enterprise Proof-of-Concept Plus programme and wish especially to acknowledge the work of Peter Balch (Analogue Information Systems Ltd) who implemented the

code from which the examples to this paper were produced. We thank the referees for their helpful comments. The photograph of the Western Gull (Larus occidentalis), taken by Dschwen, was downloaded from Wikimedia Commons and is used here under the conditions of the Free Software Foundation's GNU Free Documentation License, Version 1.2 and the Creative Commons Attribution ShareAlike 3.0 licence.

## References

[ASNA87] Takeshi Agui, M. Saito, Mazayuki Nakajima, and Y. Arai An, *An Automatic Interpolation Method of Gray-Valued Images Utilising Density Contour Lines*, Proc. Eurographics '87, 1987, pp. 405–421, ISBN 0-387-90839-0.

[Bli05] Jim Blinn, *What is a pixel?*, IEEE CG and A **25** (2005), no. 5, 82–87, ISSN 0272-1716.

[CRM96] S. Carrato, G. Ramponi, and S. Marsi, *A Simple Edge-Sensitive Image Interpolation Filter*, Proc. Int. Conf. Image Processing, 1996, ISBN, pp. 711–714.

[DHH02] David Duce, Ivan Herman, and Bob Hopgood, *SVG: Scalable Vector Graphics Tutorial*, WWW2002 Conference, Hawaii, USA, 2002, http://www.w3.org/2002/Talks/www2002-svgtut-ih/.

[Lin98] T. Lindeberg, *Edge Detection and Ridge Detection with Automatic Scale Selection*, Int. J. Comp. Vis. **30** (1998), no. 2, 117–154, ISSN 0920-5691.

[Mat75] G. Matheron, *Random Sets and Integral Geometry*, Wiley, NY, 1975, ISBN 0-471-57621-2.

[NAT83] Masayuki Nakajima, Takeshi Agui, and Masahiro Takeda, *Coding Method of Gray-Valued Image by Density Contour Lines*, Tech. Rep. I.E.C.E. Japan IE83-74 (1983), 1–6, ISBN.

[OBBT07] Alexandrina Orzan, Adrien Bousseau, Pascal Barla, and Joëlle Thollot, *Structure Preserving Manipulation of Photographs*, Proc. NPAR 2007, 2007, pp. 103–110, ISBN 978-1-59593-624-0.

[OBW+08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin, *Diffusion curves: a vector representation for smooth-shaded images*, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008) **27** (2008), no. 3, Article 92, ISSN 0730-0301.

[PB06] B. Price and W. Barrett, *Object-Based Vectorisation for Interactive Image Editing*, The Visual Computer **22** (2006), no. 9-11, 661–670, ISSN 0178-2789.

[PW06] J. Patterson and P. Willis, *Method for Image Processing and Vectorisation*, UK Patent application, 2006, GB06131199.9.

[Ree81] W. Reeves, *In-betweening for Computer Animation Utilising Moving Point Constraints*, 1981, pp. 263–269.

[Sch90] P. Schneider, ch. An Algorithm for Automatically Fitting Digitised Curves, Academic Press, 1990.

[Ser82] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982, ISBN 0126372403.

[Set99] J. Sethian, *Level Set methods and Fast Marching Methods. Second Edition*, Cambridge University Press, Cambridge, UK, 1999, ISBN 521645573.

[SLWS07] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum, *Image Vectorisation using Optimized Gradient Meshes*, ACM TOG **26** (2007), no. 3, 11–17, ISSN 0730-0301.

[Smi95] A. R. Smith, *A Pixel is Not a Little Square, A Pixel is Not a Little Square, A Pixel is Not a Little Square! (And a Voxel is Not a Little Cube)*, Technical Memo 6, Microsoft Research, 1995.

[Vin93] L. Vincent, *Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms*, IEEE Trans. Image Processing **2** (1993), no. 2, 176–201, ISSN 1057-7149.

[VWR01] Gerd Vansichem, E. Wauters, and Frank Van Reeth, *Real-Time Modelled Drawing and Manipulation of Stylised Characters in a Cartoon Animation Context*, Proc. IASTED International Conference on Computer Graphics and Imaging (CGIM 2001), 2001, pp. 44–49.

[WW67] W. Warntz and M. Woldenberg, *Concepts and Applications – Spatial order*, Harvard papers in Theoretical Geography (1967), no. 1.

**Citation**

J.W. Patterson, C.D. Taylor, and P.J. Willis, *Constructing And Rendering Vectorised Photographic Images*, Journal of Virtual Reality and Broadcasting 9(2012), no. 3, March 2012, urn:nbn:de:0009-6-32659, ISSN 1860-2037.

# Appendix: Derivation of dist()

In practice most of the following equations break down in the case of extreme conditions on the numbers of infinitesimal points inside a given $footprint()$ so we say that $\aleph(footprint(R)) >> 1$ everywhere ($\aleph$ stands for 'cardinality' or number of members of the set). There is no such restriction on S. We note also that we have slipped from using the terms R, S for borders to the regions enclosed by these borders, and this is distinguished in the following treatment.

We start by defining lines in terms of the adjacency of infinitesimal points, then define special lines, notably geodesics, which gives minimal distances. Finally we derive $dist(,)$ in terms of geodesics in the plane.

So, the adjacency or (directly) reachable relation $\leftrightarrow$ between infinitesimal points $a$ and $b$ in (any) Cartesian region R is defined using Euclidean distance

$$\delta(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

as follows:

$$a \overset{[}{R}]\leftrightarrow b \supset a \in R, b \in R - \{a\} \wedge \bar{\exists} c \in R : \delta(a,c) < \delta(a,b)$$

$$a \overset{[}{R}]\leftrightarrow b \supset \{a,b\} \subseteq R : b \overset{[}{R}]\leftrightarrow a$$

$$a \overset{[}{R}]*\leftrightarrow b \supset \{a,b\} \subseteq R, \exists c \in R : a \overset{[}{R}]\leftrightarrow c \wedge c \overset{[}{R}]*\leftrightarrow b$$

$$\text{also: } a \overset{[}{R}]*\leftrightarrow b \supset \{a,b\} \subseteq R : a \overset{[}{R}]\leftrightarrow b$$

$$\text{and } a \overset{[}{R}]*\leftrightarrow a \supset a, b \in R : a \overset{[}{R}]*\leftrightarrow b \wedge a = b$$

If $R$ is a simply connected region, then $\forall p \in R, \bar{\exists} q \in R - \{p\}, \neg\, p \overset{[}{R}]\leftrightarrow q$ In other words $p$ is reachable from every other member of $R$, and viceversa. Here the superimposed star symbol denotes Kleene closure.

A continuous line $l_R$ is a proper, partially ordered subset of a closed region $R$ defined in terms of the union of a chain of overlapping local regions $r_R(a) \subseteq l_R$ , so $l_R \subset R$ and is defined as follows (we drop the appended $R$ when it is unambiguous to do so):

$$r_l(b) = \left\{ a | \{a,b\} \subseteq l_R : a \overset{[}{l}]\leftrightarrow b \right\}$$

$$l_R = \left\{ b | \exists a \in l_R : a \overset{[}{l}]\leftrightarrow b \wedge l < \aleph(r_l(b)) \leq 3 \right\}$$

Lines are connected unordered lists of points. A member of the set will have exactly two or three adjacent points. A line may be open or closed, depending on whether it has zero or two end-points in the end-point set $P_l$ for the list $l_R$.

$$P_l = \{a | \{a\} \subseteq l_R : \aleph(r_l(a)) = 2\}$$

If $\aleph(P_l) = 0$, line $l$ is *closed*, otherwise ($\aleph(P_l) = 2$) line $l$ is *open*, with two end-points.

A closed line $l_R$ partitions a connected region $R$ (here the equivalence class defined by $inside()$) into a simple region $R'(l_R)$, the border $l_R \supset R$, and $l_R = R - footprint(R')\ iff$

$$\forall c \in R'(l_R), c\ inside R'(l_R) \supset |W_{R'}.c()| = 1$$

The border $l_R$ is thus explicitly excluded from being part of the region $R$ as only points unambiguously inside count. Care has to be taken when deciding when the border points are going to be included in a set or partition, or not.

These formulae define the length function $\langle , \rangle_{()}$ along a line in terms of Euclidean distances $\delta(,)$ between the smallest line segments. Lengths, here, imply

that points are organised as lines (as above).

$$
\begin{aligned}
&a \in l_R \supset \langle a, a \rangle_l = 0 \\
&\{a, b\} \subset l_R \wedge a \overset{[}{l}] \leftrightarrow b \supset \langle a, b \rangle_l = \delta(a, b) \\
&\{a, b, c\} \subset l_R \wedge a \overset{[}{l}] \leftrightarrow c \wedge c \overset{[}{l}]* \leftrightarrow b \supset \\
&\langle a, b \rangle_l = \delta(a, c) + \langle c, b \rangle_{l - \{a\}}
\end{aligned}
$$

The function $geo(,)$ defines a geodesic, the shortest distance between two points inside a closed, finite, irregular boundary.

$$
\begin{aligned}
&geo_R(a, b) = \\
&\left\{ c | \forall l \subset R : \langle a, c \rangle_{geo_R(a,c)} + \langle c, b \rangle_{geo_R(c,b)} \leq \langle a, b \rangle_l \right\}
\end{aligned}
$$

We can now define the (polymorphic) morphological distance function $dist(,)$ between two points $a$, $b$:

$$
dist_R(a, b) = \langle a, b \rangle_{geo_R(a,b)} \supset (geo_R(a, b) \subset R)
$$

also between a point $a$ in a closed region $R$ and a closed region $S$ enclosed by $R$:

$$
\begin{aligned}
&dist_R(a, S) = \overset{[}{b} \in S] min \langle a, b \rangle \supset \\
&\left\{ \begin{array}{l} R \; encloses \; S \wedge \exists B \in S : \forall c \in S, \\ dist_{R - footprint(S)}(a, b, ) \leq dist_{R - footprint(S)(a,c)} \end{array} \right\}
\end{aligned}
$$