

An Investigation into Image Colourisation

A Case Study in Mathematical Modelling and Scientific Computing on Image Colourisation

George Hildick-Smith

Supervised by Dr. Kathryn Gillow

May 25, 2026

© 2026 George Hildick-Smith. This work is licensed under CC BY-NC-ND 4.0, https://creativecommons.org/licenses/by-nc-nd/4.0/. Third party images and quotations remain the property of their owners.

1 Introduction

Image colourisation is the process of adding colour to a greyscale image. The origins of image colourisation can be traced back to attempts in the 19th century, when the goal was to bring historical black-and-white photographs to life. A specific example relevant to this report concerns an Italian church that was destroyed by a bomb in the Second World War. However, thankfully we had greyscale photographs of the church's artwork, and some fragments of the original painting (and its colours) remained intact as seen in Figure 1. Hence the task of colourising the photographs arose from minimal colour information.



Figure 1: *A fragmented A. Mantegna fresco (1452) damaged by bombing in the Second World War [1].*

Early algorithmic approaches to colourisation treated the problem as one of propagating sparse colour hints across an image. These methods exploited the observation that pixels with similar intensity values and spatial proximity are likely to share the same colour. Using this observation, a constrained optimisation problem over the image graph was formulated. While effective for simple scenes, such approaches struggle when the relationship between intensity and colour is non-local or when the image contains fine texture and sharp edges. More recent deep learning methods can infer plausible colours from training data alone, but they require large annotated datasets and offer limited user control over the output.

The approach taken in this report casts colourisation as a regression problem in a reproducing kernel Hilbert space (RKHS). It works as follows; given a small set of sample points at which the colour is known, find the smoothest function in a kernel-defined class that is consistent with those observations. This is a well-posed interpolation problem whose solution follows from the representer theorem. The kernel encodes both spatial proximity and intensity similarity, so the method automatically concentrates colour information near edges where abrupt transitions occur, and spreads it smoothly across uniform regions.

As an individual extension to the core method, we develop a set of heuristics for selecting the kernel parameters automatically from the image itself. We do, however, alter the original problem and develop our own strategy for placing the data points. Using heuristics removes the need for manual parameter tuning and makes the method self-contained on any input image without the need for repeated optimisation.

2 Problem Formulation

In this section, we formulate the colourisation problem precisely and fix our notation for how images and colours are represented mathematically. We define the objects involved and the mappings between them below. Please note that unless otherwise stated, all figures and images are our own creation.

2.1 Image and Colour Spaces

A digital image is typically a rectangular grid of *pixels* (derived from *pic*ture *el*ements), each assigned a colour value. We index each pixel by its integer grid coordinates (i, j) with $1 \leq i \leq H$ and $1 \leq j \leq W$, giving us $m = H \times W$ pixels. To store the pixel coordinates, we define the domain

$$\tilde{\Omega} = \{(i, j) : 1 \leq i \leq H,\ 1 \leq j \leq W\}, \quad (1)$$

such that $\tilde{\Omega} \subset \mathbb{Z}^2$. Figure 2 illustrates this, where the image is a grid of cells.

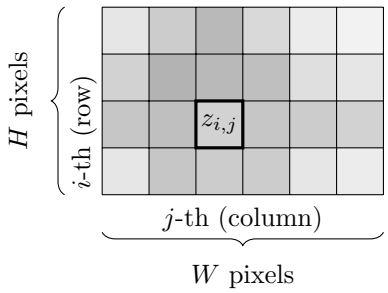


Figure 2: *A digital image as a grid of $H \times W$ pixels. Each cell $z_{i,j}$ stores a value.*

In a greyscale image, each pixel is assigned a single intensity value in $I = \{0, 1, \ldots, 255\}$, where 0 is black and 255 is white. This range arises because each pixel is given a *byte* of data to store the colour, where a byte is 8 *bits* and a bit is a single binary digit. With 8 bits, there are $2^8 = 256$ possible values, ranging from 0 to 255. We can then define a mapping

$$\tilde{\gamma} : \tilde{\Omega} \to I \quad \text{with} \quad \tilde{\gamma}(i, j) = \text{intensity of pixel at } (i, j) \in I. \quad (2)$$

Colour images are most commonly represented in the red, green, blue (RGB) colour space, although other colour spaces exist, such as hue, saturation, value (HSV) or luminance-chrominance (YCbCr) [2]. For RGB, each pixel stores three independent intensity values, one for each of the red, green, and blue channels, again drawn from I. A pixel with value (255, 0, 0) is pure red, (0, 255, 0) pure green, and (0, 0, 255) pure blue. A colour image therefore defines a mapping $\tilde{\Omega} \to I^3$. Figure 3 shows how a colour image decomposes into its three constituent channels.

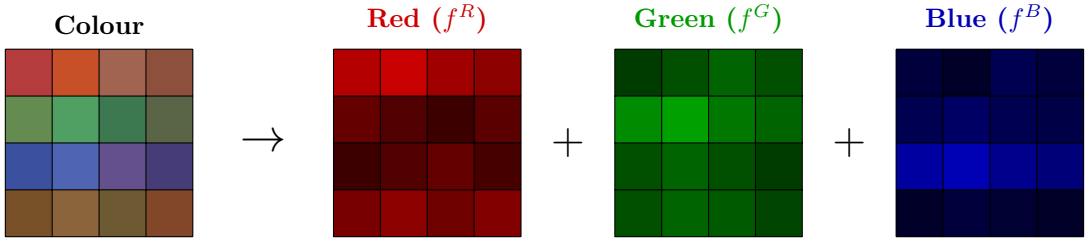


Figure 3: *Decomposition of a colour image into its three RGB channels. Each channel stores a single intensity value per pixel.*

2.2 The Colourisation Problem as Interpolation

As hinted by Figure 2, $\tilde{\Omega}$ is made up of coordinate objects that we denote as $z_{i,j} = (i, j)$. To make our notation simpler, we shall order these coordinates in a single list and hence we will have $\Omega = \{z_1, z_2, \ldots, z_m\}$ where $m = H \times W$ with bijective $g : \tilde{\Omega} \to \Omega$ such that $g(z_{i,j}) = z_{(i-1)W+j}$ i.e., we order the pixels row by row from the top left. In our problem, we are given the greyscale information at every point ($z_i \in \Omega$) and the colour information at a sparse subset of points, which we shall call $D = \{x_1, x_2, \ldots, x_n\} \subset \Omega$, with $n \ll m$. For this ordering, we define $\gamma : \Omega \to I$ such that $\gamma(z_i) = \tilde{\gamma}(g^{-1}(z_i))$ that stores the greyscale information. Now let $f : D \to I^3$ encode the colour information at the points in D. We shall use the channel decomposition to simplify our problem and treat each channel $s \in \{R, G, B\}$ as a separate scalar interpolation problem. Hence, we will have three separate mappings $f^R, f^G, f^B : D \to I$. As each solves the same problem in their respective channels, we will reference them as f^s where s is the colour channel. The goal of the problem is to find a mapping $F = (F^R, F^G, F^B) : \Omega \to I^3$ such that $F(x_i) \approx f(x_i)$ for all $x_i \in D$.

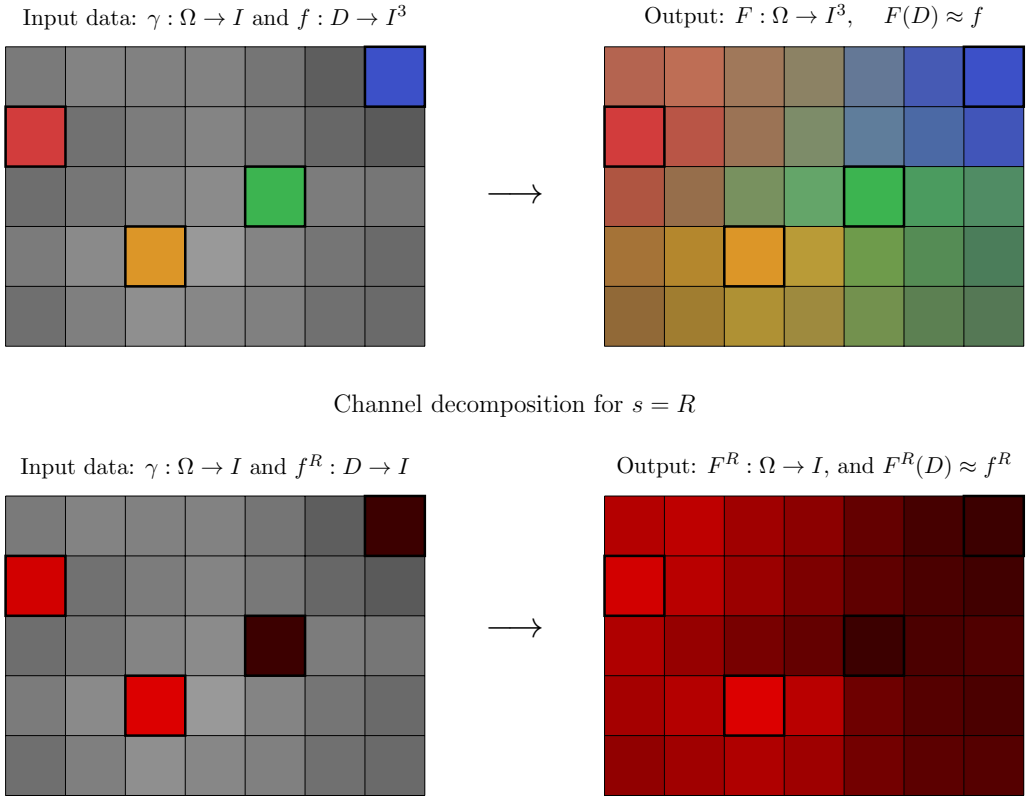


Figure 4: *Top row: the full colourisation problem with given γ and sparse colour data f. Bottom row: the same problem restricted to one channel, $s = R$.*

3 Reproducing Kernel Hilbert Spaces

To recover the full colour image from sparse data, we need a methodical way to extend a function defined on D to all of Ω. This section develops the mathematical framework that makes this possible, namely RKHSs.

3.1 Hilbert Spaces and Reproducing Kernels

To find a function $F^s : \Omega \to I$ that agrees with the known data f^s on D and varies smoothly elsewhere for each channel s, we have to search some function space. If the space is too small we risk missing good solutions; if it is too large, finding any becomes intractable. We choose a RKHS that will strike the right balance.

We begin by recalling that a *Hilbert space* $\mathcal{H}$ is a vector space of functions with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, which measures how alike two functions are, and is *complete*[1]. It has an induced norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ which measures the "size" or complexity of f. What makes a RKHS special and a good choice is the existence of *reproducing kernels*. These are functions $k : \Omega \times \Omega \to \mathbb{R}$ with the property that for each fixed $x \in \Omega$, the function $k(\cdot, x)$ belongs to $\mathcal{H}$, and the evaluation of any $f \in \mathcal{H}$ at a point x can be recovered by taking an inner product,

$$f(x) = \langle f,\, k(\cdot, x) \rangle_{\mathcal{H}} \quad \text{for all } f \in \mathcal{H},\ x \in \Omega. \tag{3}$$

This is called the *reproducing property*. Intuitively, $k(\cdot, x)$ acts as a "probe", and the inner product of f with this probe extracts the value of f at x. For this property to hold, the kernel k must be symmetric and positive semi-definite (PSD), meaning the $n \times n$ matrix K_D with entries $(K_D)_{ij} = k(x_i, x_j)$ satisfies $\mathbf{c}^\top K_D \mathbf{c} \geq 0$ for all $\mathbf{c} \in \mathbb{R}^n$ and any choice of $\mathbf{x}$. The *Moore-Aronszajn theorem* then guarantees that every such kernel determines a unique RKHS, so by choosing k we are implicitly choosing the function space in which we search for F^s [3].

As an example, consider the kernel $k(x, y) = \min(x, y)$ for $x, y \geq 0$. The corresponding RKHS has the inner product $\langle f, g \rangle = \int_0^\infty f'(t)\, g'(t)\, \mathrm{d}t$. We can verify the reproducing property directly for $f(x) = x$ by fixing y such that $\langle f, k(\cdot, y) \rangle = \int_0^\infty \mathbf{1}_{[t \leq y]}\, \mathrm{d}t = y = f(y)$. It is worth emphasising that this example is somewhat special in that the inner product has an explicit closed form. In general, the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is never computed directly, and its expression need not be known. The kernel k is the object we specify and work with in practice.

[1] Completeness means that any sequence of functions that gets arbitrarily close to one another in the sense of this inner product converges to a limit that also lies in $\mathcal{H}$.

3.2 The Representer Theorem

Rather than searching over all functions in $\mathcal{H}$ (in general, an infinite-dimensional space), we want to find F^s by minimising a cost that balances how well it fits the data against how complex it is. A natural choice is the regularised least-squares problem

$$\min_{F^s \in \mathcal{H}} \left(\sum_{x_i \in D} \left(F^s(x_i) - f^s(x_i) \right)^2 + \delta \|F^s\|_{\mathcal{H}}^2 \right), \tag{4}$$

where $\delta > 0$ is a *regularisation parameter*. The first term penalises deviation from the known colour data on D; the second term penalises functions that are large in the norm of $\mathcal{H}$, preventing overfitting. Despite the problem being posed over an infinite-dimensional space, the *representer theorem* tells us the minimiser has a finite and explicit form [3].

Theorem 3.1 (Representer Theorem)**.** *The minimiser of Equation (4) in $\mathcal{H}$ exists, is unique, and takes the form*

$$F^s(z) = \sum_{i=1}^{n} \alpha_i^s \, k(z, x_i), \tag{5}$$

for some coefficients $\alpha_1^s, \ldots, \alpha_n^s \in \mathbb{R}$.

This is a remarkable reduction, as no matter how large or complex $\mathcal{H}$ is, the solution always lies in the n-dimensional subspace spanned by the kernel functions $\{k(\cdot, x_i)\}_{i=1}^n$ centred at the data points. Each $k(\cdot, x_i)$ is a bump-shaped function centred at x_i, and F^s is a weighted sum of these bumps. The problem of finding a function over all of Ω reduces to finding n real numbers, $\alpha_1^s, \ldots, \alpha_n^s$.

3.3 The RKHS Colourisation Formula

Following the RKHS approach of [4], we derive an expression for F^s. By the representer theorem, the solution to the regularised interpolation problem takes the form of Equation (5) where k is the kernel function and $\alpha_i^s \in \mathbb{R}$ are the coefficients to be determined. Evaluating this at the data points $\{x_j\}_{j=1}^n$ and imposing the regularised fit condition gives

$$(K_D + \delta I_n)\, \boldsymbol{\alpha}^s = \boldsymbol{f}^s, \tag{6}$$

where $(K_D)_{ij} = k(x_i, x_j)$ is the $n \times n$ kernel matrix evaluated on D, $\boldsymbol{f}^s = (f^s(x_1), \ldots, f^s(x_n))^\top$ is the vector of given values in channel s, $\delta > 0$ is a regularisation parameter, and I_n is the $n \times n$ identity matrix. Equation (6) is a linear system that can be solved for $\boldsymbol{\alpha}^s$. Once we have $\boldsymbol{\alpha}^s$, we can evaluate F^s at any point $z_i \in \Omega$ using Equation (5) and the problem is solved.

4 Kernel Design

In this section, we describe the choice of kernel k used in the colourisation formula. From Theorem 3.1, the kernel determines how strongly each known data point $x_i \in D$ influences the prediction at any query location $z_i \in \Omega$. There are two key properties a good kernel should satisfy. First, pixels that are spatially close to a data point should receive more influence than those far away. However, a kernel that considers only spatial proximity will bleed colour across sharp boundaries, as illustrated in Figure 5. Secondly, we want pixels with similar greyscale intensity to have a strong influence, even when not adjacent, as these are likely to have a similar colour.

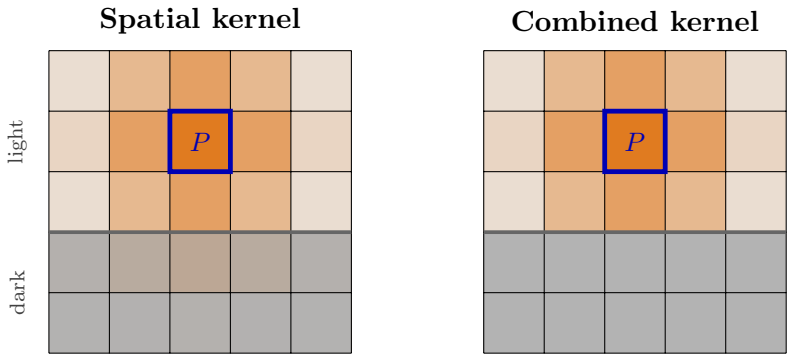


Figure 5: *Sketched influence of a pixel P for the two kernels. Orange shading indicates kernel magnitude at each cell. The spatial kernel (left) bleeds into the dark region, while the combined spatial-intensity kernel (right) confines its influence to pixels of similar greyscale.*

4.1 Radial Basis Functions

To achieve this first property, we introduce a radial basis function (RBF) kernel, which depends only on the distance between two points and some parameters. A general RBF kernel is

$$k(x, y) = \phi\left(\frac{\|x - y\|^p}{\sigma^p}\right), \tag{7}$$

for some function $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$, length-scale parameter $\sigma > 0$ and $p > 0$ which control how quickly the kernel decays and the shape of the decay, respectively. As mentioned in Section 3.1, for the RKHS machinery to apply, k must be PSD. Bochner's theorem [5] states that a continuous function ϕ yields a PSD kernel if and only if it is the Fourier transform of a non-negative measure on $\mathbb{R}^d$. We consider the Gaussian and Wendland RBFs here and show the Gaussian satisfies this condition. Proving that Wendland functions are PSD is more involved and does not use Bochner's theorem. Instead, Wendland used the Askey scheme of truncated power functions that preserve positive-definiteness [5], but proving this is outside the scope of this report.

4.2 The Gaussian RBF

The Gaussian function is

$$\phi_G(r) = \exp\bigl(-r^2\bigr), \tag{8}$$

giving the kernel $k_G(x, y) = \exp\bigl(-\|x - y\|^{2p}/\sigma^{2p}\bigr)$. For $p = 1$ this is PSD by Bochner's theorem, since the Fourier transform of $r \mapsto e^{-r^2/\sigma^2}$ is $\xi \mapsto \sigma\sqrt{\pi}\, e^{-\sigma^2\xi^2/4}$, which is non-negative. The result extends to $\mathbb{R}^d$ by separability, since $e^{-\|x-y\|^2/\sigma^2} = \prod_{i=1}^{d} e^{-(x_i-y_i)^2/\sigma^2}$ is a product of 1D PSD kernels, and a pointwise product of PSD kernels is PSD. For $0 < p \leq 1$ the kernel remains PSD, which is shown in [5]. Hence k_G is PSD for any $\sigma > 0$ and $0 < p \leq 1$. For $p > 1$, the kernel is not guaranteed to be PSD, so we restrict to $p \leq 1$. Its main limitation is global support: ϕ_G never reaches exactly zero, so every data point influences every other in principle, making K_D dense and expensive to factorise.

4.3 The Wendland RBF

The Wendland function is

$$\phi_W(r) = \bigl(1 - r\bigr)_+^4 (4r + 1), \tag{9}$$

where $(t)_+ = \max(t, 0)$, meaning it has compact support on $[0, 1]$. This function is exactly zero for all $r \geq 1$, so the kernel vanishes whenever $\|x - y\| \geq \sigma$, regardless of p. The matrix K_D is therefore sparse (when σ is small), which brings significant computational savings for large images. The trade-off is that the Wendland function only achieves C^2 smoothness, compared to the Gaussian C^∞, though C^2 is usually sufficient for piecewise-smooth images. We can see both functions in Figure 6.

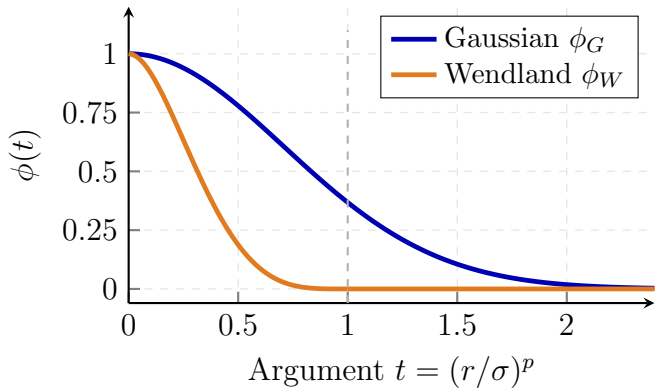


Figure 6: *Plots of the Gaussian and Wendland radial basis functions against their argument $t = (r/\sigma)^p$. The Wendland function vanishes exactly for $t \geq 1$.*

4.4 The Combined Spatial-Intensity Kernel

As we mentioned at the beginning of Section 4, a purely spatial RBF kernel propagates colour in rings around each data point regardless of any edges in the image. To prevent colour from crossing intensity boundaries, we incorporate the greyscale map γ into the kernel. This is because pixels with similar greyscale intensity are likely to belong to the same region of the image and share a colour. We define the combined kernel as a product of a spatial term and an intensity term,

$$k(x, y) = \phi\left(\frac{\|x - y\|}{\sigma_1 \, d}\right) \cdot \phi\left(\frac{|\gamma(x) - \gamma(y)|^p}{\sigma_2 \cdot 255^{\,p}}\right), \tag{10}$$

where $d = \sqrt{W^2 + H^2}$ is the image diagonal and 255 is the maximum greyscale value. These are in the denominators to normalise each term. The parameters σ_1 and σ_2 set the spatial and intensity length-scales as fractions of the image size and greyscale range, respectively, following [4]. The exponent $p \geq 0$ controls sensitivity to intensity differences, with larger p concentrating the kernel more tightly on pixels of nearly identical intensity. This product structure ensures the kernel is large only when two pixels are both spatially close and share a similar greyscale.

Figure 7 shows the practical consequence of the choice between Gaussian and Wendland profiles at the matrix level. The Gaussian K_D is much denser than the Wendland K_D, which is nearly diagonal with only spatially adjacent, same-region point pairs contributing non-zero entries.

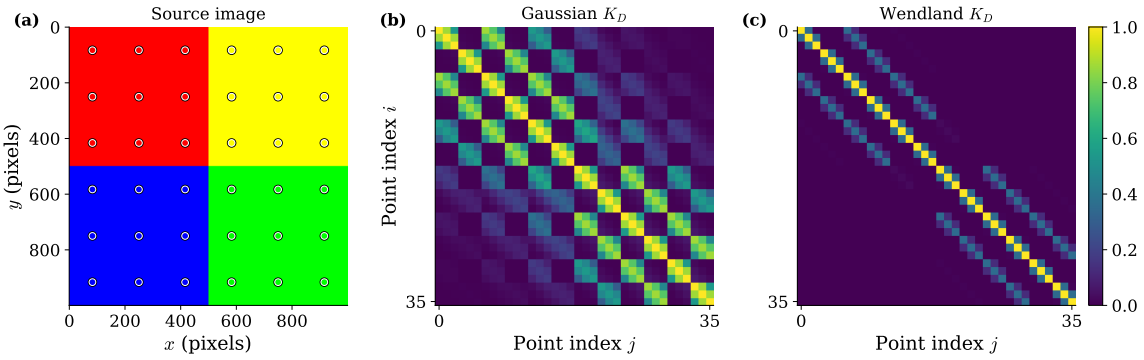


Figure 7: ***(a)*** *A test image with a* 6×6 *regular grid of colour-known data points that make up* D. ***(b)*** *Matrix* K_D *for the Gaussian kernel.* ***(c)*** *Matrix* K_D *for the Wendland kernel.*

It is clear to see from Equation (10) that $k(x, x) = 1$ for all x, hence the diagonal entries of K_D are all 1 and appear yellow for both kernels.

5 The Algorithm and Implementation

To test our algorithm, we start with images in full colour, from which we compute the synthetic input data, namely the greyscale image and the colour information at the points in D. This allows comparison against the ground truth and evaluation of colourisation quality; these results then guide our parameter and sampling choices. The algorithm is given in Figure 8, and each section will be described in turn.

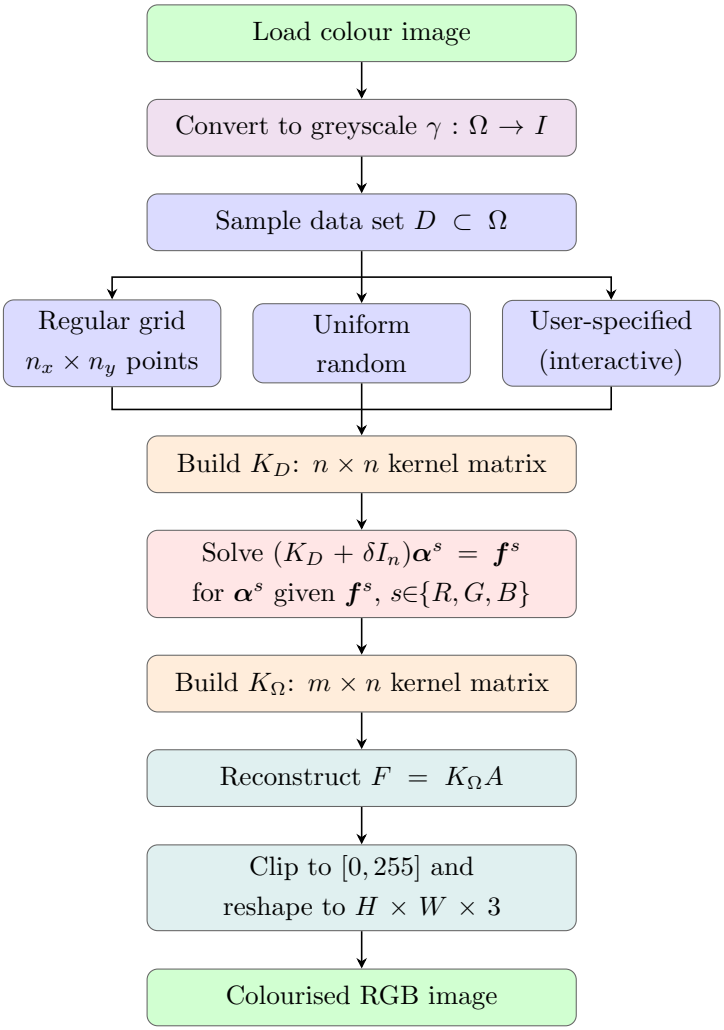


Figure 8: *Full colourisation pipeline: green input/output; violet preprocessing; blue sampling strategies; orange kernel construction; red linear solve; teal reconstruction.*

5.1 Converting to Greyscale

The greyscale image is computed from the RGB image using the standard luminosity method, which is a weighted sum of the three channels. The weights are chosen to approximate human perception of brightness in each channel. The formula is

$$\gamma(x) = \lfloor 0.299\, R(x) + 0.587\, G(x) + 0.114\, B(x) \rceil \in I, \tag{11}$$

5.2 Sampling Strategies for the Data Set D

We implement three strategies. A *regular grid* of $n_x \times n_y$ evenly spaced interior points, as shown in Figure 7 (a), is simple to control but may miss features in images with irregular structure. *Uniform random* sampling removes this bias, but some areas may be undersampled. Finally, *user-specified* sampling lets the user click directly on the image, enabling deliberate placement at region boundaries or intentional omission of a region to test extrapolation.

5.3 Building the Kernel Matrices

Two kernel matrices appear in the algorithm. The $n \times n$ matrix K_D encodes pairwise similarities among the data points in D, as illustrated in Figure 7 (b) and (c):

$$(K_D)_{ij} = \phi\left(\frac{\|x_i - x_j\|}{\sigma_1\, d}\right) \phi\left(\frac{|\gamma(x_i) - \gamma(x_j)|^p}{\sigma_2 \cdot 255^p}\right), \quad 1 \le i, j \le n. \tag{12}$$

The $m \times n$ matrix K_Ω is computed on every pixel $z_i \in \Omega$:

$$(K_\Omega)_{ij} = \phi\left(\frac{\|z_i - x_j\|}{\sigma_1\, d}\right) \phi\left(\frac{|\gamma(z_i) - \gamma(x_j)|^p}{\sigma_2 \cdot 255^p}\right), \quad 1 \le i \le m,\ 1 \le j \le n. \tag{13}$$

Both use the combined spatial-intensity kernel from Equation (10).

5.4 Solving for Coefficients

With K_D computed, the coefficient vector $\boldsymbol{\alpha}^s$ for each channel $s \in \{R, G, B\}$ is found by solving the regularised system from Equation (6),

$$(K_D + \delta I_n)\, \boldsymbol{\alpha}^s = \boldsymbol{f}^s, \tag{14}$$

where $\boldsymbol{f}^s \in \mathbb{R}^n$ is the vector of known channel values at the data points in D. This is solved using an LU factorisation, and as the three channels share the same coefficient matrix, K_D, the factorisation is reused across all three, which means we do not have to solve the full system three times.

5.5 Reconstructing the Colourised Image

Once the coefficient matrix $A = [\boldsymbol{\alpha}^R\ \boldsymbol{\alpha}^G\ \boldsymbol{\alpha}^B] \in \mathbb{R}^{n \times 3}$ is known, the full colourised image is obtained by a single matrix product

$$F = K_\Omega\, A \in \mathbb{R}^{m \times 3}, \tag{15}$$

where row i of F holds the predicted (R, G, B) triple at pixel $z_i \in \Omega$. The result is reshaped to $H \times W \times 3$ and clipped to $[0, 255]$ if necessary.

6 Initial Results and Analysis

With the algorithm fully specified, we now apply it to a test image to see how well it performs depending on the parameters σ_1, σ_2, p, and δ. Figure 9 **(b)** shows the input to the algorithm, while **(c)** shows the output. Compared to the original image in **(a)**, the colourisation captures the overall structure and colour distribution, but looks inaccurate near the boundaries. This is a boundary effect: points near the image edges have fewer nearby data points and so receive less colour information.

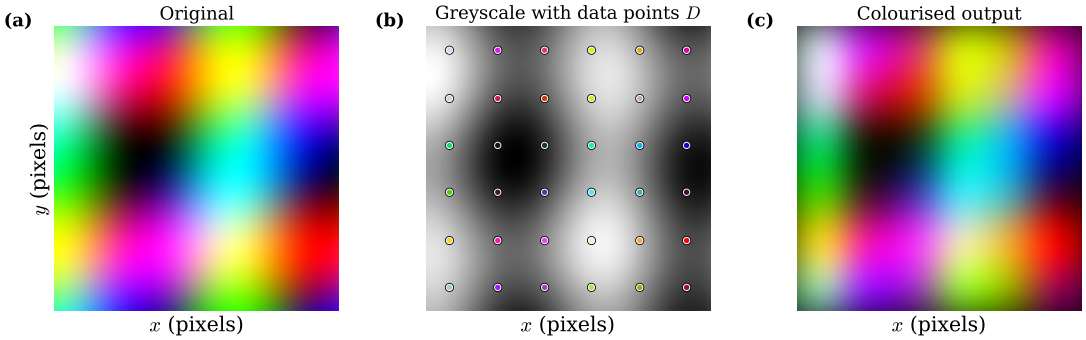


Figure 9: *Gaussian colourisation pipeline applied to a test image.* ***(a)*** *Original colour image.* ***(b)*** *Greyscale with the* 6×6 *grid of colour-known data points.* ***(c)*** *Colourised output produced by the RKHS regression.* $\sigma_1 = 0.1, \sigma_2 = 1, p = 1, \delta = 0.1$.

To try it yourself, please follow this link[2].

For small σ_1, the spatial kernel decays quickly, so each point only influences its immediate neighbourhood, producing isolated coloured spots at the grid locations in Figure 10. However, when σ_1 increases, the radius of influence grows. When σ_2 is small, the intensity kernel is highly selective and colour propagates only to pixels whose greyscale value closely matches the data point, depositing colour along curves with the same intensity and producing the thin bands visible in the first column. Increasing σ_2 relaxes this constraint, allowing colour to spread regardless of intensity difference and blurring the contours into the smooth fields seen in the top right.

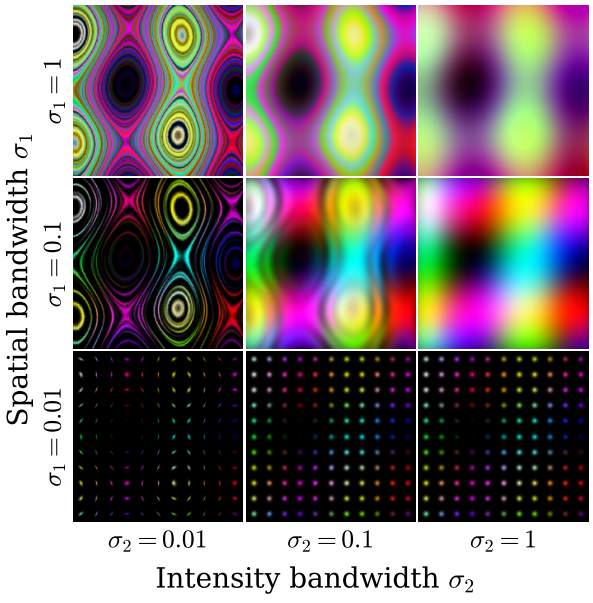


Figure 10: *Colourised output for* (σ_1, σ_2) *pairs from* $\{0.01, 0.1, 1, 10\}$ *with* $p = 1$, $\delta = 0.1$ *and Gaussian kernel.*

[2] https://g323hs.github.io/rkhs-colourisation/

7 Quantitative Evaluation

Figures 9 and 10 give an intuitive sense of how well the algorithm worked, but to compare parameter settings or kernel types objectively, we need scalar metrics that compare the original image A with the output B. We consider two of these metrics.

7.1 Pixel-wise Frobenius Norm

The Frobenius norm treats every pixel and channel equally, and is given by

$$\mathcal{E}_F(A,B) \;=\; \|A-B\|_F \;=\; \sqrt{\sum_{z_i\in\Omega}\sum_{s\in\{R,G,B\}}\big(A^s(z_i)-B^s(z_i)\big)^2}, \tag{16}$$

A value of zero means perfect reconstruction, and larger values indicate greater deviation. However, $\mathcal{E}_F$ grows with image resolution, since doubling the side lengths quadruples the number of pixels and inflates the norm even when the per-pixel error is unchanged, meaning scores are not directly comparable across images of different sizes. It is also sensitive to errors throughout the image, so a small uniformly distributed error and a large localised error can give the same score even though they look very different perceptually.

7.2 Structural Similarity Index

The Structural Similarity Index Measure (SSIM) addresses the perceptual limitation of $\mathcal{E}_F$ by comparing the luminance, contrast and structure of two images locally. Let $\mathcal{W}(z_i)$ denote a small Gaussian-weighted window[3] centred at z_i with normalised weights $\{w_{z_j}\}_{z_j\in\mathcal{W}(z_i)}$. The local mean, variance and covariance of channel s at pixel z_i are then

$$\mu_{A,s}(z_i)=\sum_{z_j\in\mathcal{W}(z_i)} w_{z_j}A^s(z_j), \qquad \sigma^2_{A,s}(z_i)=\sum_{z_j\in\mathcal{W}(z_i)} w_{z_j}\big(A^s(z_j)-\mu_{A,s}(z_i)\big)^2, \tag{17}$$

$$\sigma_{AB,s}(z_i)=\sum_{z_j\in\mathcal{W}(z_i)} w_{z_j}\big(A^s(z_j)-\mu_{A,s}(z_i)\big)\big(B^s(z_j)-\mu_{B,s}(z_i)\big). \tag{18}$$

and $\mu_{B,s}(z_i)$, $\sigma^2_{B,s}(z_i)$ are defined analogously. The local similarity at z_i for channel s is

$$\mathrm{SSIM}_s(z_i)=\frac{\big(2\mu_{A,s}(z_i)\,\mu_{B,s}(z_i)+c_1\big)\big(2\sigma_{AB,s}(z_i)+c_2\big)}{\big(\mu^2_{A,s}(z_i)+\mu^2_{B,s}(z_i)+c_1\big)\big(\sigma^2_{A,s}(z_i)+\sigma^2_{B,s}(z_i)+c_2\big)}, \tag{19}$$

where c_1 and c_2 are small constants that stabilise the ratio when the denominator is near zero. The global score averages over an interior region[4] $\Omega'\subseteq\Omega$ and all three channels,

$$\mathrm{SSIM}(A,B)=\frac{1}{3|\Omega'|}\sum_{s\,\in\,\{R,G,B\}}\sum_{z_i\,\in\,\Omega'}\mathrm{SSIM}_s(z_i), \tag{20}$$

taking values in $[-1,1]$, with 1 indicating identical images. An animation of the colourisation process as n increases can be seen here[5].

[3] We use an 11×11 window with a Gaussian of standard deviation $\sigma=1.5$, matching [6].

[4] Pixels within $\lfloor w/2\rfloor$ of the image boundary are excluded, where w is the window side length, to avoid artefacts where the window would extend beyond the image.

[5] https://youtu.be/N6waK8awbe4

8 Parameter Heuristics (Individual Extension)

At this point, we slightly alter the original problem. Suppose we are now able to choose where our data points in D are located, given a budget of n points rather than a fixed set. This immediately raises the question of where to place them.

For this new problem, we try to optimise the parameters and the location of the data points in D. However, the scope of this adjusted optimisation problem becomes intractable even for very small images. For instance, fixing a budget of $n = 10$ points on a 10×10 image already gives $\binom{100}{10} \approx 1.7 \times 10^{13}$ possible choices of D alone, before the kernel parameters are considered. If each evaluation took just 1 ms, an exhaustive search would require roughly 540 years of computation time. There are optimisation techniques that can be used to reduce the search space; however, they are still computationally expensive. Therefore, we propose a heuristic approach to point placement parameter selection that gives good rough estimates of the optimal parameters without the need for an exhaustive search. From this section onwards, we use only the Gaussian kernel with $p = 0.5$ and $\delta = 2 \times 10^{-4}$.

8.1 Image Preprocessing and Scaling

For large images, building $K_\Omega \in \mathbb{R}^{m \times n}$, where $m = W \times H$ is the total pixel count and n is the number of data points, becomes prohibitively expensive in both memory and compute. If $W \times H$ exceeds a user-chosen pixel budget P, we will first downsample the image by a uniform scale factor $s = \sqrt{P/(W \times H)}$, giving a resized image of dimensions $\lfloor sW \rfloor \times \lfloor sH \rfloor$ using LANCZOS resampling [7]. All subsequent steps, edge detection, point placement, and parameter estimation, then operate on the scaled image, so that n reflects the actual resolution at which the algorithm runs rather than the original file size. In our experiments, we use $P = 160{,}000$. Figure 11 shows this applied to a real photograph.

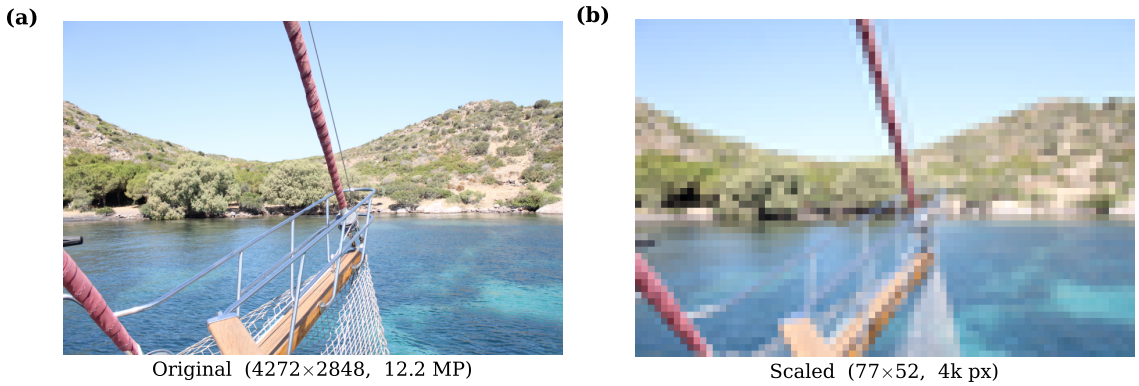


Figure 11: ***(a)*** *Original image at* 4272×2848 *(12.2 MP).* ***(b)*** *LANCZOS downsampled to* 77×52 *(4k px) using* $P = 4{,}000$*, scale factor* $s = 0.018$*.*

8.2 Edge Detection

Finding where edges are in the image and where the image is smooth versus detailed will help inform where we should place our data points. To get an estimate of the edge locations, we use a simple gradient-based method. For each pixel $x \in \Omega$, we calculate the gradient magnitude of the greyscale,

$$g(x) = \|\nabla I(x)\|, \tag{21}$$

approximated using finite differences. We classify pixels with $g(x) > \tau$ as edge pixels for a threshold τ chosen by Otsu's method [8], which splits the population into a smooth class $C_0(\tau) = \{x \mid g(x) \leq \tau\}$ and an edge class $C_1(\tau) = \{x \mid g(x) > \tau\}$ with weights $\omega_k(\tau) = |C_k(\tau)|/|\Omega|$ and means $\mu_k(\tau)$, where $k \in \{0, 1\}$. The between-class variance is

$$\sigma_B^2(\tau) = \omega_0(\tau)\,\omega_1(\tau)\,\big(\mu_1(\tau) - \mu_0(\tau)\big)^2, \tag{22}$$

and Otsu's threshold is the value that maximises this quantity,

$$\tau^* = \arg\max_{\tau}\ \sigma_B^2(\tau). \tag{23}$$

Maximising $\sigma_B^2(\tau)$ pushes the two class means as far apart as possible while keeping each class tightly concentrated. A practical difficulty arises when a small sharp boundary dominates the gradient histogram, and its outlier spike inflates τ^*, causing most true edges to fall below the threshold and be discarded. To guard against this, g is clipped at a chosen percentile before Otsu's method is applied, compressing the outlier spike without altering the bulk of the distribution. Furthermore, τ^* is clipped to ensure between 10% and 90% of pixels are classified as edges. To suppress isolated noise pixels and close small gaps in edge bands, a Gaussian blur is applied to the raw binary mask with standard deviation 1 px and the result is re-thresholded at 0.5. We write $\mathcal{S} \subseteq \Omega$ for the set of smoothed edge pixels and $\mathcal{S}^c = \Omega \setminus \mathcal{S}$ for the smooth interior. Figure 12 shows each stage of the pipeline.

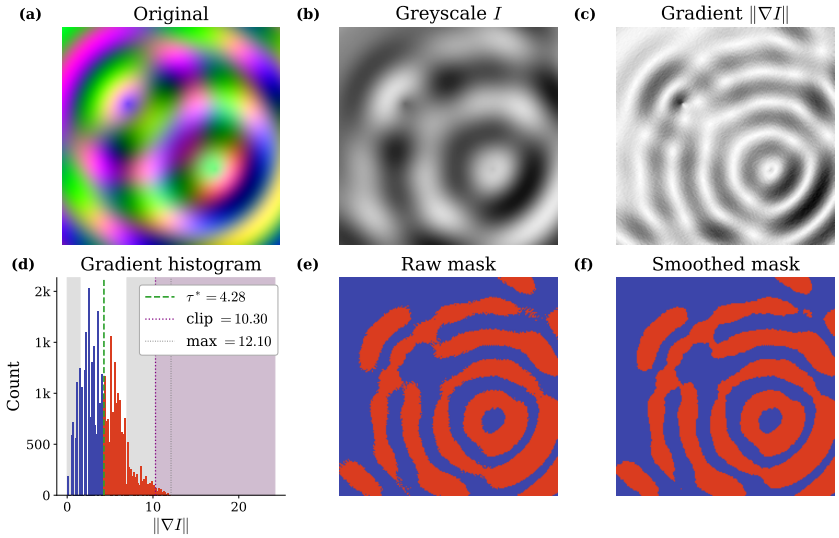


Figure 12: ***(a)*** *Original image.* ***(b)*** *Greyscale I.* ***(c)*** *Gradient magnitude $\|\nabla I\|$.* ***(d)*** *Gradient histogram.* ***(e)*** *Raw binary edge mask.* ***(f)*** *Smoothed mask, $\mathcal{S}$ (red), $\mathcal{S}^c$ (blue).*

8.3 Point Placement

Given the scaled image of dimensions $W \times H$, we set the total number of sample points to

$$n = \lfloor 0.01 \cdot WH \rceil \,, \tag{24}$$

that is, 1% of all pixels (this can be changed later). This keeps $K_D \in \mathbb{R}^{n \times n}$ small enough to factorise efficiently while providing enough points to reconstruct broad colour regions.

To decide *where* we should place the n points, we use a technique we shall call edge-adaptive Poisson-disk sampling. In standard Poisson-disk sampling (also known as Blue noise), a minimum separation distance r is enforced between every pair of accepted points, so the final set covers the domain without either large gaps or clumps of points [9]. For an explanation, diagram and flowchart of this algorithm, see Appendix A. In Figure 13, we can see that Blue noise is a combination of the other two methods. This is ideal for our problem as it gives us good coverage of the image while avoiding the bias that a grid introduces.

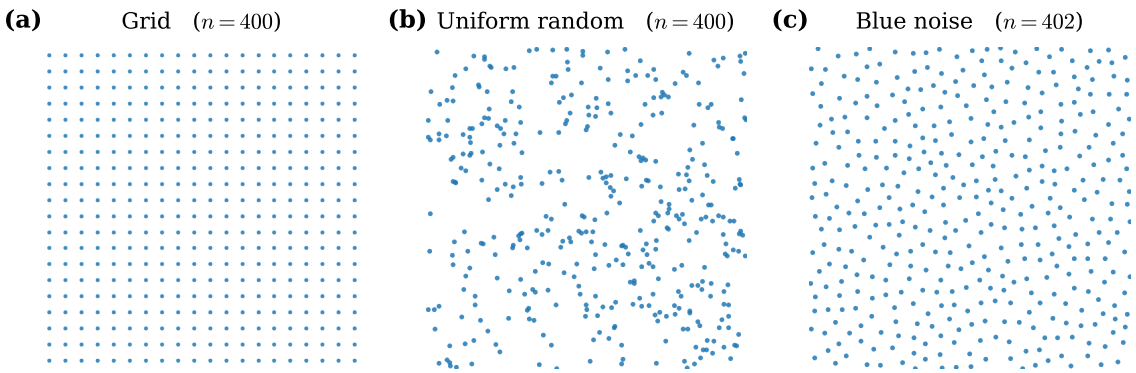


Figure 13: *Comparison of sampling strategies for $n = 400$ target points.*

We may note that the Blue noise contains 402 points rather than exactly $n = 400$. This is because the algorithm terminates when no further point can be placed at least r from all existing points, so the final count is determined by the packing geometry. To choose r for a given target n, we use the equation

$$nr^2 \approx c \approx 0.67, \tag{25}$$

which is the empirically determined packing density of this method. Figure 14 shows the distribution of point counts over 1000 runs with different seeds. We computed this for a range of r values and averaged the c they found.

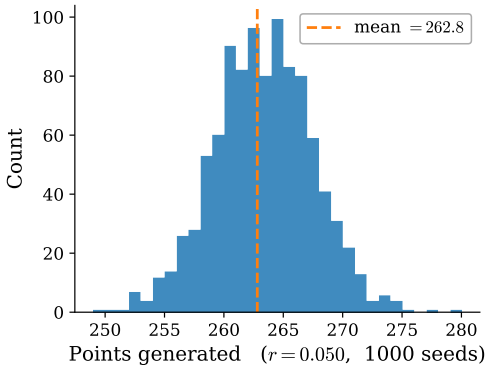


Figure 14: *Distribution of Blue-noise point count over 1000 seeds with $r = 0.05$. Dashed line is the mean.*

Now that we know how to choose r for a given n, we remind ourselves of Figure 12 **(f)**, which shows the edge mask $\mathcal{S}$ and smooth interior $\mathcal{S}^c$. From this analysis, we know that the red regions are detailed, so we propose sampling at a density ξ times higher in edge regions than in smooth ones. Given a budget of n points, we can split this into $n_{\mathcal{S}}$ points for the edge set $\mathcal{S}$, and $n_{\mathcal{S}^c} = n - n_{\mathcal{S}}$ for the smooth interior $\mathcal{S}^c$, hence

$$n_{\mathcal{S}} = \frac{\xi\,|\mathcal{S}|}{\xi\,|\mathcal{S}| + |\mathcal{S}^c|} \cdot n, \quad \text{and} \quad n_{\mathcal{S}^c} = \frac{|\mathcal{S}^c|}{\xi\,|\mathcal{S}| + |\mathcal{S}^c|} \cdot n. \tag{26}$$

With $n_{\mathcal{S}}$ and $n_{\mathcal{S}^c}$, we can compute the corresponding $r_{\mathcal{S}}$ and $r_{\mathcal{S}^c}$ for each region using Equation (25).

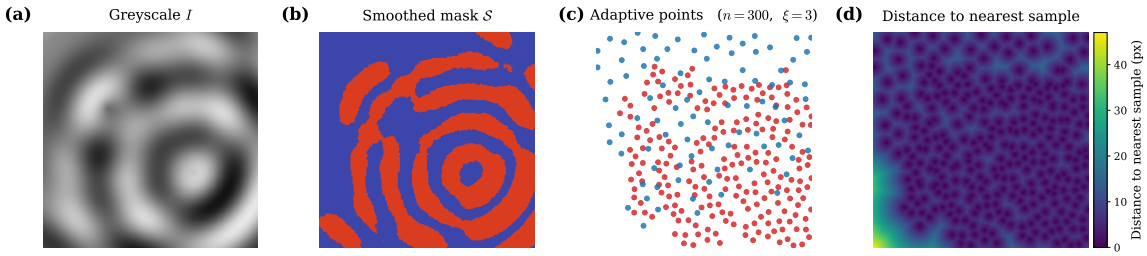


Figure 15: *Edge-adaptive Poisson-disk sampling. (a) greyscale I; (b) the smoothed edge mask $\mathcal{S}$; (c) blue-noise set with $\xi = 3$; (d) the Euclidean distance from each pixel to its nearest sample point. We only accept samples that also lie in the corresponding region.*

From Figure 15 **(c)**, we can see how the density is higher in complex regions. However, we also see that there are some regions with no points nearby, which is not ideal, as just because the greyscale is smooth does not mean the colour is. To fix this, we reserve 2% of the points for a final spatial fill step where, for each point reserved, we calculate the pixel that is furthest from any existing point and add that point to our set. We shall call this step the 'spatial fill step'. Hence, we apply Equation (26) and multiply each count by 98%.

The algorithm may not yield exactly $n_{\mathcal{S}}$ or $n_{\mathcal{S}^c}$ points per region, so we run Poisson-disk sampling in each region until the target count is reached. If the algorithm terminates early, the shortfall is absorbed into the spatial fill step. We can see the effect of this in Figure 16. The green dots show where the points have been added; these tend to be on the edge. The grey points are not exactly the same as the blue and red points in Figure 15 **(c)** because the spatial fill step means we take fewer points from the Poisson-disk sampling.

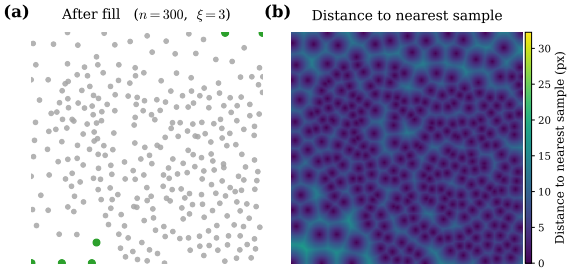


Figure 16: *The effect of the spatial fill step.*

8.4 Spatial Bandwidth σ_1

The parameter σ_1 controls how far colour information propagates from each sample point. We set σ_1 so that the kernel $\phi(r) = \exp(-r^2)$, evaluated at the *worst-covered* pixel of Ω, is at least some confidence level $\beta_1 \in (0, 1)$. We define the *fill distance*

$$h_D = \max_{z \in \Omega} \min_{x_i \in D} \|z - x_i\|, \tag{27}$$

which is the distance from the pixel furthest from any sample, to that pixel's nearest sample. The spatial kernel weight that pixel receives is $\phi(h_D/(\sigma_1\, d)) = \exp\!\left(-\left(\frac{h_D}{\sigma_1\, d}\right)^2\right)$, where d is the diagonal. Requiring this to be at least β_1 gives $\left(\frac{h_D}{\sigma_1\, d}\right)^2 \le -\ln \beta_1$, so

$$\sigma_1^* = \frac{h_D}{d\,\sqrt{-\ln \beta_1}}. \tag{28}$$

In our implemented algorithm, we choose $\beta_1 = 0.95$, so the denominator is roughly $d\sqrt{-\ln 0.95} \approx 0.226\, d$, and $\sigma_1^* \approx 4.4\, h_D/d$. Using the running example in Figures 12, 15 and 16 of radial interference, we generate Figure 17 which shows how the kernel weight at h_D varies with σ_1. We calculate σ_1^* as the value at which the curve crosses β_1.

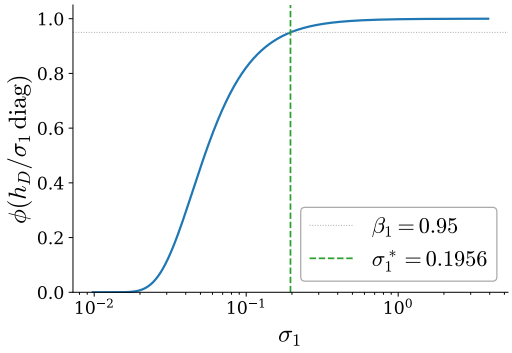


Figure 17: *Kernel weight $\phi(h_D/\sigma_1 d)$ as a function of σ_1.*

8.5 Intensity Bandwidth σ_2

The parameter σ_2 controls how different (in greyscale) a pixel can be from a sample point and still receive significant colour from it. Setting σ_2 too large makes every sample influence every pixel regardless of intensity, washing out colour edges. Setting it too small means pixels receive no colour at all from samples with a slightly different intensity. The right scale is the average greyscale difference between a pixel and its nearest sample, which should be small as nearby pixels share similar intensities. For each pixel $z \in \Omega$, let $x_{j(z)}$ be its spatially nearest sample. We define the *spatial-conditional intensity gap*

$$h_\gamma = \max_{z \in \Omega} |\gamma(z) - \gamma(x_{j(z)})| \tag{29}$$

as the worst-case greyscale difference between any pixel and the sample that will most naturally influence it. Applying the same coverage argument as for σ_1 where we require the intensity factor of the kernel to be at least β_2 at h_γ, we get

$$\sigma_2^* = \frac{h_\gamma}{255\,(-\ln \beta_2)^{1/(2p)}}. \tag{30}$$

Figure 18 shows $|\gamma(z) - \gamma(x_{j(z)})|$ at every pixel for the interference-pattern example. Bright regions are far in greyscale from their nearest sample, while dark regions are already well matched. By choosing h_γ as the maximum of this map, we ensure that even the worst-matched pixel receives enough colour from its nearest sample. In our implementation, we set $\beta_2 = 0.95$ and $p = 0.5$, so $\sigma_2^* \approx 0.27\, h_\gamma$.

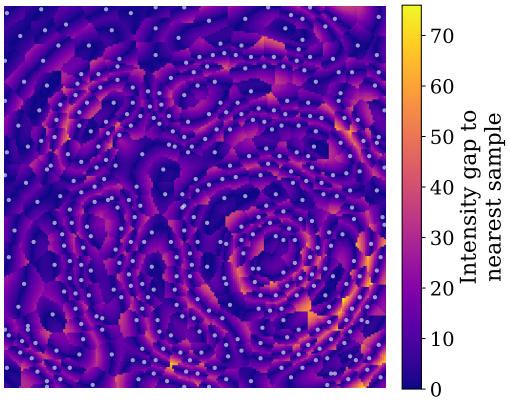


Figure 18: *Spatial-conditional intensity gap* $|\gamma(z) - \gamma(x_{j(z)})|$ *at each pixel.*

8.6 Results

Figure 19 shows the full colourisation pipeline applied to the radial interference pattern introduced in Figure 12. Starting from the original colour image, we extract the greyscale channel I, compute the smoothed Otsu edge mask to identify the edge region $\mathcal{S}$ and smooth interior $\mathcal{S}^c$, then place $n = 400$ colour-known sample points using the adaptive Poisson-disk scheme with spatial fill. The heuristic bandwidths $\sigma_1^* \approx 0.196$ and $\sigma_2^* \approx 10.64$ are computed from the fill distance and intensity gap, and the system $(K_D + \delta I_n)\boldsymbol{\alpha}^s = \boldsymbol{f}^s$ is solved for each channel with $p = 0.5$ and $\delta = 2 \times 10^{-4}$ with the Gaussian kernel. The colourised output achieves a Frobenius error of $\mathcal{E}_F = 2156.96$ and an SSIM of 0.9717. The Frobenius error is hard to interpret in isolation, but the high SSIM indicates that the output is perceptually very close to the original. Indeed, by eye, the colourisation looks very good.

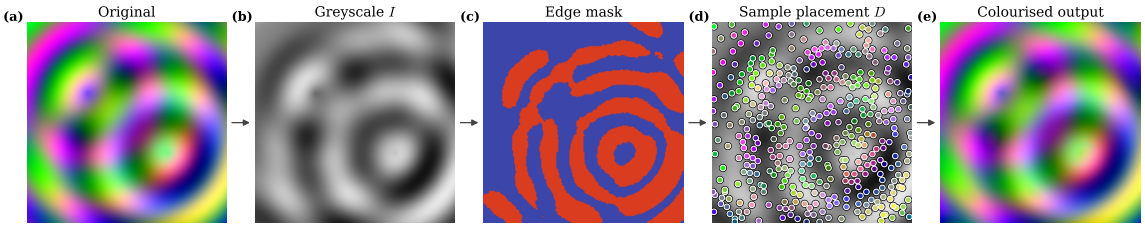


Figure 19: *The complete colourisation pipeline applied to the radial interference pattern.* ***(a)*** *Original colour image.* ***(b)*** *Greyscale* I*.* ***(c)*** *Smoothed edge mask (*$\mathcal{S}$ *red,* $\mathcal{S}^c$ *blue).* ***(d)*** *Adaptive sample placement* D *(*$n = 400$ *points, colours show the known RGB values).* ***(e)*** *Colourised output.*

To demonstrate that the algorithm generalises to a variety of image types, further examples are shown in Appendix B.

We encourage the reader to try it themselves[6].

[6] https://g323hs.github.io/rkhs-colourisation/

9 Conclusion

In this report, we have developed a mathematically grounded approach to image colourisation from sparse colour data. We formulated the problem as interpolation in a reproducing kernel Hilbert space, which reduces the task of searching an infinite-dimensional space to finding a colour function over Ω and then to solving a finite linear system via the representer theorem. The solution $F^s = \sum_{i=1}^{n} \alpha_i^s k(\cdot, x_i)$ is a weighted sum of kernel bumps centred at the data points, with coefficients obtained by solving $(K_D + \delta I_n)\boldsymbol{\alpha}^s = \boldsymbol{f}^s$ for each colour channel.

The kernel combines spatial proximity and greyscale similarity, with bandwidth parameters σ_1 and σ_2 controlling how far colour information can propagate in space and in intensity respectively. The parameter grid in Figure 10 illustrates the sensitivity of the output to these choices. Small σ_1 confines colour to the immediate neighbourhood of each sample, while small σ_2 restricts propagation to pixels of nearly identical intensity. Making good parameter choices is therefore essential, which motivates our individual extension.

The individual extension augmented the original problem by allowing us to choose where to place our budget of n sample points. To do this, we first apply Otsu thresholding on the gradient magnitude to partition the image into an edge region $\mathcal{S}$ and a smooth interior $\mathcal{S}^c$. Then, an edge-adaptive Poisson-disk scheme concentrates more samples near edges where the image changes rapidly. A greedy spatial fill step eliminates any large voids left by the Poisson-disk pass. This gives us a set of sample points D that covers the image well without clumping, and is denser in complex regions where more information is needed. Finally, we automatically tune the kernel parameters to the image and the point set we have just chosen. The spatial bandwidth σ_1^* follows from the fill distance h_D via the coverage condition $\phi(h_D/\sigma_1^* d) \geq \beta_1$, giving $\sigma_1^* = h_D/(d\sqrt{-\ln \beta_1})$. The intensity bandwidth σ_2^* is derived analogously from the spatial-conditional intensity gap h_γ. This performs remarkably well on all kinds of images, as shown in Appendix B.

10 Future Work

Building on the work done in this report, we could reformulate the problem in the YCbCr colour space, which would reduce the regression from three channels to two, since the Y channel is already known (it is exactly the greyscale image I). This would improve computational efficiency as the number of independent linear systems to solve would be reduced. Alternatively, we could investigate automatic parameter tuning without the added freedom of choosing point locations. Returning to the original fresco problem, some whole colour fragments survive intact. We could investigate how best to exploit these intact fragments to improve the colourisation. Unlike the isolated points assumed throughout this report, a surviving fragment provides dense colour information over a continuous patch of the image, and incorporating such region-level data directly into the kernel regression framework would be a natural extension.

11 References

[1] M. Fornasier, *Nonlinear projection digital image inpainting and restoration methods*, 2004. arXiv: math/0412112. [Online]. Available: https://arxiv.org/abs/math/0412112.

[2] Wikipedia contributors, *RGB color spaces — Wikipedia, the free encyclopedia*, [Online; accessed 15 May 2026], 2024. [Online]. Available: https://en.wikipedia.org/wiki/RGB_color_spaces.

[3] V. I. Paulsen and M. Raghupathi, *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces* (Cambridge Studies in Advanced Mathematics). Cambridge University Press, 2016, vol. 152.

[4] M. Ha Quang, S. H. Kang, and T. M. Le, "Image and video colorization using vector-valued reproducing kernel hilbert spaces," *Journal of Mathematical Imaging and Vision*, vol. 37, no. 1, pp. 49–65, 2010.

[5] H. Wendland, *Scattered Data Approximation* (Cambridge Monographs on Applied and Computational Mathematics). Cambridge University Press, 2004, vol. 17.

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[7] C. E. Duchon, "Lanczos filtering in one and two dimensions," *Journal of Applied Meteorology*, vol. 18, no. 8, pp. 1016–1022, 1979.

[8] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. DOI: 10.1109/TSMC.1979.4310076.

[9] R. Bridson, "Fast poisson disk sampling in arbitrary dimensions," *ACM SIGGRAPH 2007 Sketches*, p. 22, 2007. [Online]. Available: https://www.cs.ubc.ca/~rbridson/docs/bridson-siggraph07-poissondisk.pdf.

[10] M. C. Escher, *Waterfall*, Lithograph, The Art Institute of Chicago, accession no. 1987.199.4, 1961.

[11] R. Penrose, "The role of aesthetics in pure and applied mathematical research," *Bulletin of the Institute of Mathematics and its Applications*, vol. 10, no. 2, pp. 266–271, 1974.

[12] L. S. Penrose and R. Penrose, "Impossible objects: A special type of visual illusion," *British Journal of Psychology*, vol. 49, no. 1, pp. 31–33, 1958. DOI: 10.1111/j.2044-8295.1958.tb00634.x.

Appendix

A Blue noise Sampling

Blue noise sampling produces point sets that look random yet are also roughly evenly spaced, as illustrated in Figure 20. This means that both low-frequency clumping and high-frequency lattice structures are suppressed. Bridson's algorithm [9] achieves this in $O(n)$ time by combining a spatial acceleration structure with an active-list rejection, as shown in Figure 22. Suppose our domain is $[0,1]^2$, and we want points roughly $0 < r < 1$ distance apart. First, we divide the domain into a grid of square cells with side $r/\sqrt{2}$, chosen so that any cell contains at most one sample point (the diagonal will be length r). A seed point is placed at random and added to an *active list*. At each iteration, an active point x is selected at random and up to $k = 30$ candidate points c are drawn uniformly from the annulus $r < \|c - x\| < 2r$ and tested for validity, meaning c must lie in the annulus and also be farther than r from every existing point. This check reduces to examining only the neighbouring grid cells. If a valid candidate is found, it is accepted and appended to the active list. Otherwise, x is removed from the active list. The algorithm terminates when the active list is empty, at which point every empty region of the domain is (with high probability) less than r away from any existing point. Figure 21 illustrates this.

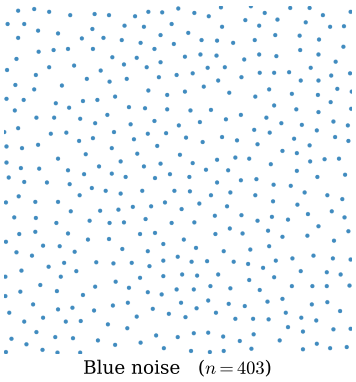


Figure 20: *Blue noise (Poisson-disk) sample points.*

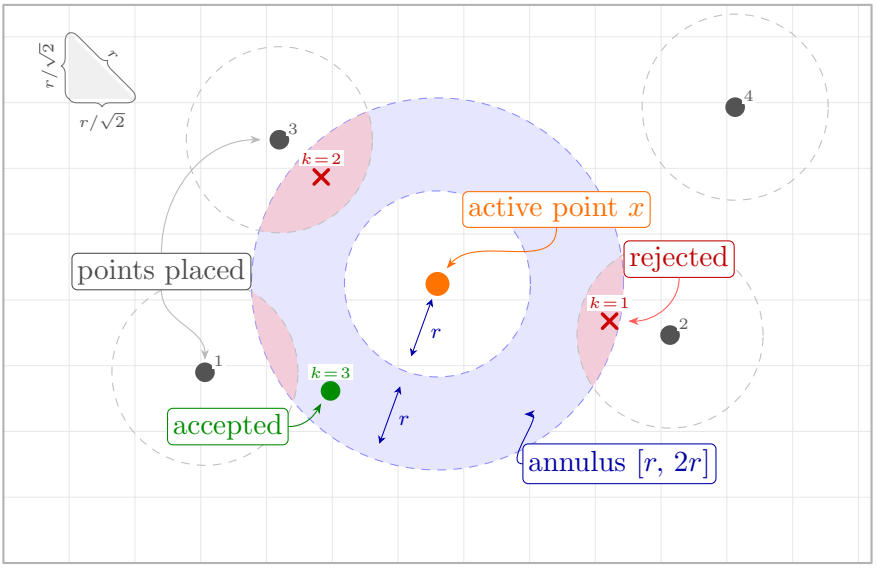


Figure 21: *One step of Bridson's Poisson-disk algorithm. The orange point is active; grey points (1–4) are already placed, each surrounded by a dashed exclusion zone of radius r. Candidates are drawn from the blue annulus $[r, 2r]$: attempts $k = 1$ and $k = 2$ fall inside existing exclusion zones (red) and are rejected; attempt $k = 3$ is accepted.*

Figure 22 gives a complete step-by-step view of the algorithm. The outer loop continues until the active list is exhausted, at which point the domain is fully packed to within distance r of every sample point. The inner loop draws up to $k = 30$ candidates per active point before declaring it exhausted and removing it.

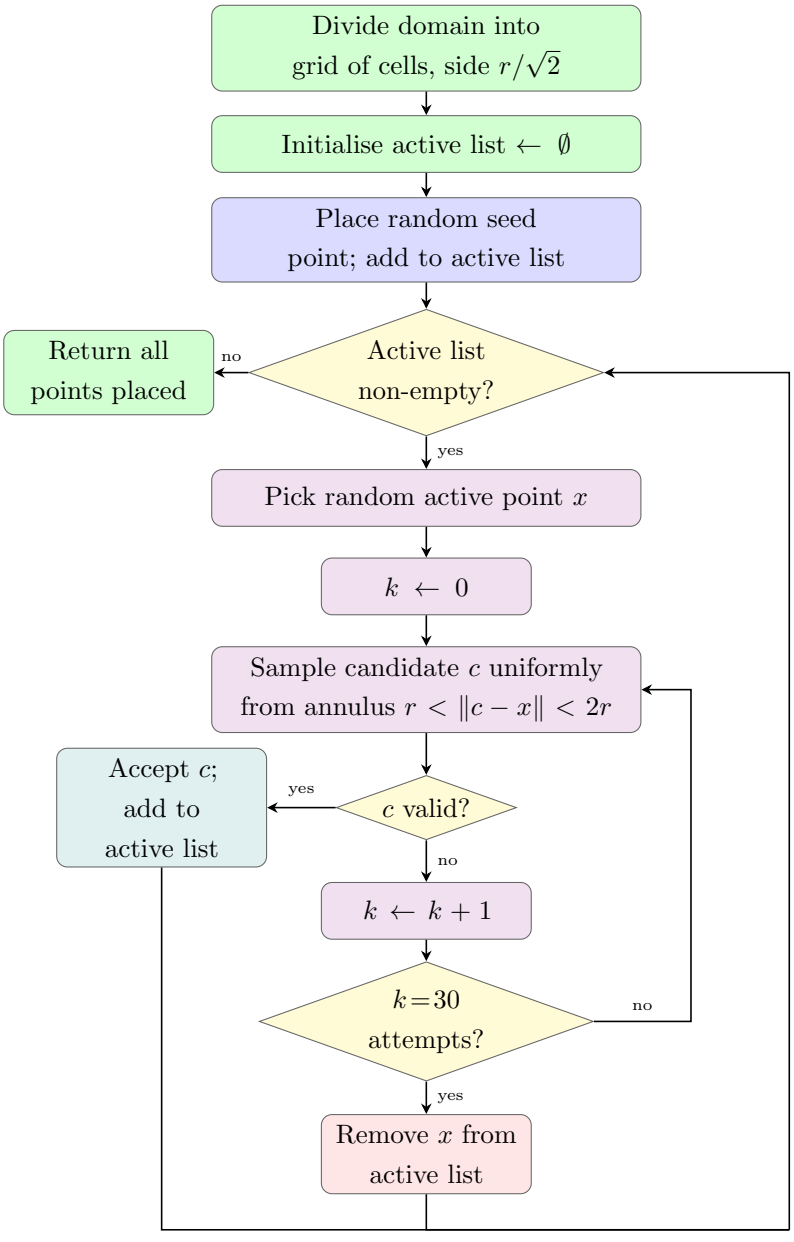


Figure 22: *Bridson's algorithm [9]: green initialisation and output; blue seed placement; violet inner loop body; yellow decisions; teal accepted candidate; red rejection and removal.*

B Pipeline Gallery

Figure 23 shows the complete colourisation pipeline applied to five images of different character, from a real photograph to smooth gradients. In every case, the edge mask, adaptive sampling, and heuristic bandwidths are computed automatically from the image without any manual tuning, and the colourised output recovers the original colours.

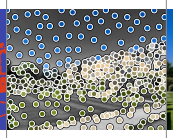
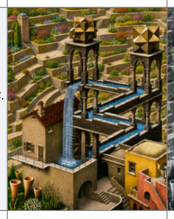
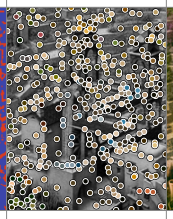
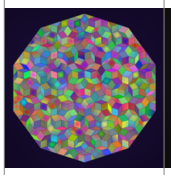
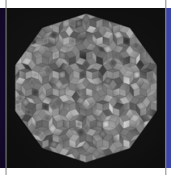
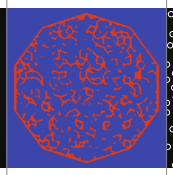
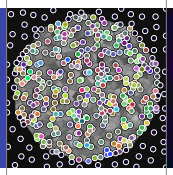
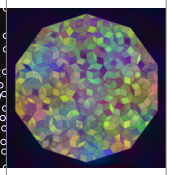
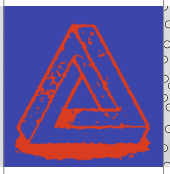
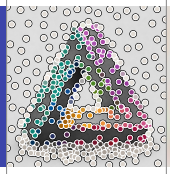
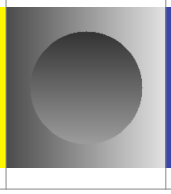
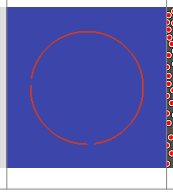
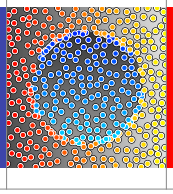
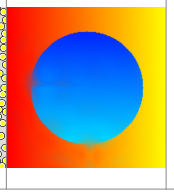
Image	Original	Greyscale I	Edge mask	Sample placement D	Colourised output	$\|\cdot\|_F$	SSIM
Worcester College, Oxford						3508.4	0.929
Coloured M.C. Escher waterfall						5880.8	0.898
Penrose tiling						11081.9	0.570
Penrose triangle						3403.6	0.897
Gradient disk						3164.9	0.907

Figure 23: *Each row shows one image through the five steps: original colour, greyscale I, edge mask, sample placement D ($n = 1\%$ of pixels) with density ratio $\xi = 10.0$, and colourised output. The five test images are Worcester College Oxford, M.C. Escher's Waterfall [10] (colourised by ChatGPT), a Penrose tiling [11], a Penrose triangle [12] (rendered by ChatGPT), and a synthetic gradient disc.*

The photograph of Worcester College performs well, as it contains large regions of smoothly varying colour. The Penrose tiling is more challenging due to its high-frequency pattern, where a sample point must fall within a given tile for that tile's colour to be recovered accurately. The gradient disc struggles at the six and nine o'clock positions on the circle, where the greyscale values on either side of the boundary are nearly identical, causing colour to bleed across the edge. This is a fundamental limitation of the kernel regression approach, which relies on greyscale contrast to guide colour propagation. An approach that additionally exploited gradient direction or curvature information might be able to overcome this.

C Glossary

Acronym	**Definition**
HSV	hue, saturation, value
MSE	mean squared error
PSD	positive semi-definite
RBF	radial basis function
RGB	red, green, blue
RKHS	reproducing kernel Hilbert space
SSIM	Structural Similarity Index Measure
YCbCr	luminance-chrominance

Table 1: *Glossary of acronyms used in this report.*