# Interpolation Techniques Applied to Color Images

Mark A. Fitch

## 1 Motivation

Consider the extremely popular World Wide Web (WWW or web). Almost every web "page" contains images for all the world to see. Consider also, that no one wants the display pictures with defects. This is just one of the motivations for techniques for removing defects from digital images. Other uses of digital image editing are print and video advertising and movie animation and special effects.

## 2 Problem Description

The most obvious approach to removing defects in images is to paint over the defective portions. In degree to ensure that a re-touched image looks natural, the colors used to repaint must closely match those in the rest of the picture. Additionally, the edges of the edited area must smoothly transition into adjoining areas. Rather than studying art at a local conservatory, numerical interpolation techniques can be used to generate appropriate colors.

Essentially, interpolating the colors is an improved version of copying a non-defective portion of the image over the defect. The problem with copying is that the lighting effect may not be correct. Interpolating the colors using nearby points will duplicate the lighting effect of the sample. For this to work well, there must be sufficient points about the defect to produce an accurate approximation of the natural color gradient. If the area being edited is on the edge of an object, then the ability to copy the lighting effect is particularly helpful. Because the domain and range of an interpolation function can be stretched, the sampled portion of the picture can be stretched to cover extra area. This effectively extends the lighting scheme to cover the entire object.

Another advantage of interpolation schemes is smoother edge transitions. If a section is simply copied, the difference between colors at the edges may be noticeable. Interpolation schemes that force the interpolating function to match the data at the sample points will have edge transitions as smooth as the original picture.

## 3 Method of Solution

First, it is important to understand how color is represented in digital images. One of the most common methods is Red/Green/Blue (RGB). Under the RGB scheme, a color is represented by the intensity values of the three basic color components of light: red, green, and blue. Thus color can be represented as a 3-tuple. The intensities are measured from 0 to some max value $M$. This max value $M$ is mapped to the highest intensity possible on the display device (usually a monitor). Higher values of $M$ indicate smaller possible gradations in the color scheme. Finer gradations will make it easier to produce smooth transitions. The most common maximum value today is 255. It is important to note that some digital imaging systems have a high $M$ value but do not allow all possible colors to be used at once.

In general, polynomial interpolation takes a set of knots $(x_i, f_i)$ and produces a polynomial that approximates a function $f$, where the nodes $x_i$ are considered to be elements of the domain space of $f$, and $f_i = f(x_i)$ for all $i$. A variety of additional conditions can be applied to effect the polynomial chosen. In the most general setting for the image applications, the function $f$ has as its domain a set of coordinates and has the set of colors as its range, i.e., $f : Z^2_{mxn} \rightarrow Z^3_M$.

In degree to simplify implementation, the following assumptions can be made. First, the domain of an interpolating functions can be expressed as a linearly increasing set such as a column or row. Thus the domain can be thought of as a finite subset of the integers. This assumption is necessary to use classical polynomial interpolation techniques.

Note that the domain could be take as any curve that can be parameterized by a single parameter. Thus diagonal lines and circles can be used as the domain (sample points) if this is useful. Remember, however, that if the curve will be used to fill an area, it may not be completely covered due to the discretization of the curve coordinates from the parametric equations. For example, if concentric circles are used, then the center of the circles will not be covered by any circle with positive radius. Further the inner circles will be very small making the color gradient appear rougher.

For any curve chosen as the domain, this linear interpolation approach does not directly guarantee smooth color transitions perpendicular to the curve. For example, if an area is edited by interpolating the colors in columns, then the colors along the rows are not guaranteed to transition smoothly. However, if the interpolations are good approximations of the original color, and the function is not being stretched to cover a larger area than the sample area, then this problem should be minimal.

A second assumption about the interpolation schemes is that the function can always be expressed as

$$f(x_i) = \begin{bmatrix} f_1(x_i) \\ f_2(x_i) \\ f_3(x_i) \end{bmatrix},$$

that is, there is no dependence between terms. As a result three interpolations, one for each color component, can be used. This assumption seems intuitively reasonable. Color happens to be represented by three components, but the intensity of one component is not dependent on the intensity of another component.

Any implementation of a color editing scheme using interpolation must also handle the transition from the continuous range of the interpolating polynomial to the finite range of the color space. First the intensity values from the interpolating polynomial must be rounded to integers. The simplest approach is simply to use the floor or ceiling functions. If the image is using sufficiently many colors, the slight difference between the colors produced by choosing the floor or ceiling will not be noticeable.

In cases where fewer colors are available or cause major speed decreases, a more intelligent method would be to consider the adjoining points. Choosing the floor for local max points, the ceiling for local min points, and defaulting to either for all other points will minimize the range of intensities for each component. This could help reduce the number of colors used.

Another potential problem is negative intensity values or values above the maximum value $M$. Such values could be caused by one color component reaching 0 or $M$. For example, any gradient fading to black, such as a shadow would produce, will have intensity values near 0. Since the interpolating function is a polynomial it may generate slightly negative values. However, negative intensity values of large magnitude and values far above $M$ are unlikely under the interpolation schemes considered.

The easiest approach to handling illegal values is simply to replace all overly high and low values with the $M$ or zero respectively. Because the values are likely only slightly above or below, no loss of a smooth gradation will be caused. In cases where the extra values are of extreme importance, the max and min values of the interpolating polynomial over the needed domain can be found and the colors of the entire image scaled to make these max and min values $M$ and zero respectively.

Two interpolation schemes were considered. The first is the cubic spline with not-a-knot condition. Two properties of cubic splines are useful for image editing applications. First, matching the actual function (color) values at the data points will help the edited area match the unedited areas in color. Second, the continuous first derivative of the interpolating polynomial ensures that no sharp changes in color occur, i.e., a smooth looking color gradient is produced. The not-a-knot condition is not important in itself. The clamped cubic spline might be useful in ensuring that the edited portion fits smoothly into the rest of the image, but since the function, hence its derivative, is not known, this condition cannot be used. Likewise the Hermite cubic spline method cannot be used. The natural cubic spline condition assumes that the color will be changing linearly outside of the sample which is not a fair assumption. Thus the not-a-knot condition was chosen.

The second approach is a least square error approximation. Note that the error minimized is the component-wise error, that is, a polynomial with least square error is fit to each color component. The error under the standard Euclidean norm on the 3x1 color vectors is not necessarily minimized. Based on the assumption that the color components are independent, this should not cause major differences. So long as this simpler approach produces acceptable results, modifying the approach to minimize the vector error is unnecessary.

A potential problem with the least square error approach is non-smooth edge transitions. Since one edge of the edited area will likely be a data point in all likelihood then the difference between the color produced by the interpolating polynomials and the color of the neighboring point will be small. However the other edge might not be a sampled point. This might occur if points are sampled uniformly. For example if every tenth point is sampled from 1 to 90 then 81 is the last sampled point, but 81-90 are also changed. Thus, especially if a polynomial of high degree is used, the intensity values of the last points may be quickly departing from the desired gradient. An easy way to fix this problem is to force both end points to be sample points.

## 4 Investigation

Note that all numeric manipulations were performed using Matlab. Matlab's standard *spline* and *polyfit* functions were used to find the cubic spline and least square error interpolating polynomials respectively. Pictures were stored in the Unix *ppm* format using 24 bit color ($M = 255$). All code and examples can be found in the directory mfitch/klassen/mth860/project.

A picture of a coke can against a white paper background was used to test the ability to produce useful approximations from a relatively small number of points. In the original digital image, the lighting of the left portion of the background did not match the lighting of the right portion of the background.

To produce an uniform background effect, the entire image to the left of the can was replaced with colors produced by interpolated color gradients. The image is 120 pixels in height and every 10th pixel from 1 to 111 was sampled. Both interpolation methods produced colors that closely matched the color gradient on the right. For the least squares method polynomials of degree 3, 6, and 9 were tried. Using 24 bit color little difference existed. Under 8 bit color ($M = 255$ but not all colors can be displayed), the difference was noticeable, but all still matched the right hand side closely. Based on the results of this test it appears that both cubic spline and least square error polynomial interpolation methods can work well with a limited number of points.

A picture of a face was used to test the smoothness of edge transitions. The forehead and hair areas were replaced column by column by the output of the interpolating polynomials generated from data points of the forehead region. The result is to produce a picture of a bald version of the face. The lowest sample point for each column was the same, so that the edge of the edited area is a straight line just above the eyebrows. The cubic spline method did not produce a noticeable edge in the re-touched picture. A least square error polynomial of degree 3 produced almost identical results. Polynomials of degree 6 and 9 produced using the least square method did create noticeable edges.

Stretching the color gradient to cover more area than was sampled worked as well. The spline and 3rd degree least square error polynomials produced a realistic looking bald head. The 6th degree polynomial produced overly dark regions from the smaller samples at the left and right hand side of head. The 9th degree polynomial produced non-flesh colors at the top of the head. Thus it appears that these polynomials produced too much variation. This is likely the result of stretching very small samples (roughly 5 points).

The distortions produced by the higher degree polynomials caused the rows to have sudden color transitions. However, the spline and 3rd degree least square error polynomial did not exhibit this problem. These results were independent of the number of colors used (24 bit or 8 bit color). This seems to indicate that sampling and editing images in lines does not produce rough color gradients perpendicular to the sampling curve.

# 5 Conclusion

Currently, many jobs exist that require removing defects of color digital images. One approach to the editing is to use polynomial interpolation techniques to produce color gradients to replace the defects. Based on the test performed, it appears that interpolation techniques can be successfully used under many conditions.