# A FAMILY OF NEW ALGORITHMS FOR SOFT FILLING

## (Extended Abstract) †

*Kenneth P. Fishkin*

*Brian A. Barsky*

Berkeley Computer Graphics Laboratory
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720
U.S.A.

ABSTRACT

*Soft filling* algorithms change the colour of an anti-aliased region, while maintaining the anti-aliasing of the region. The two published algorithms for soft filling work only if the foreground region is anti-aliased against a black background. This paper presents three new algorithms. The first fills against an *arbitrary single-coloured* background and is faster than the published algorithms on a pixel-by-pixel basis for an RGB frame buffer; the second fills against a background composed of *two* arbitrary colours; and the third fills against a background composed of *three* arbitrary colours. As the power of the algorithms increases, so do the number of assumptions they make, and the computational cost.

*KEYWORDS:* Algorithms, Colour, Filling, Painting systems.

### 1. Introduction

A *region* is a group of connected pixels in a frame buffer. This region possesses a certain colour, which may have been created by direct user "painting", or as the result of a rendering algorithm. The problem of changing the displayed colour of this rendered region is known as *filling*. The filling algorithm has no "world knowledge" of the characteristics of the region; it is given only the old colour of the region, the new colour desired for it, and one "seed point" known to be inside. Given this information, the algorithm traverses the region, changing the old region colour to the new.

If the region is anti-aliased, or was painted with a non-opaque brush, the process is known as *soft filling*, and the region is termed *soft-edged*. If the region is defined by a boundary, its colour is changed by *boundary-filling* algorithms; if the region is defined by its interior values, its colour is changed by *flood-filling* algorithms. In this paper we focus on flood filling, although our algorithms work equally well for boundary filling.

We refer to the region whose colour is to be changed as the *foreground*, and the colour of that region as the *foreground colour*. The set of underlying colours that surround the foreground are the *background* colours. For a soft-edged region, the foreground is considered to include those pixels that are partial blends of the foreground and background colour(s). Conceptually, each pixel only *partially* contains the region colour; only this part of the pixel's colour should be changed.

† The complete paper will appear in *Proceedings of SIGGRAPH '84*.[1]

This work was supported in part by the Semiconductor Research Corporation under grant number 82-11-008 and the National Science Foundation under grant number ECS-8204381.

A filling algorithm can be decomposed into four components:

- The *START procedure*: processing done to initialize the algorithm.
- The *propagation method*: how the algorithm decides where to search for the borders of the region.
- The *INSIDE procedure*: the processing done when a pixel is read to decide whether it is to be filled. (Smith's[5] GET procedure is contained in this).
- The *SET procedure*: the processing done to change the colour of a given pixel.

The two published algorithms for soft filling[3,5] work only if the foreground region is rendered against a black background. In this paper, a family of new algorithms is presented. The published algorithms are shown to be computationally less efficient special cases of the first new algorithm, which fills against an *arbitrary* single-coloured background. The second fills against a background composed of *two* different colours, and the third against a background composed of *three* different colours. As the power of the algorithms increases, so does the cost of computation.

## 2. Existing Algorithms

### 2.1. Interior Fill

When the region is not anti-aliased, a simple, quick algorithm known as *interior fill*[5] can be used. This algorithm successfully fills against an arbitrary number of arbitrary background colours. However, it assumes hard-edged, non-anti-aliased regions.

### 2.2. Tint Fill

When the region is soft-edged, the problem is much more difficult. The edge of the region (or even the interior, if the region is shadowed or has specular reflection) gradually blurs into the other colours. Deciding when a pixel is INSIDE, and if so to what degree it is composed of which background colours, is a more difficult matter.

The most popular algorithm for soft filling is Smith's *tint fill*.[5] In a later paper,[6] Smith presented an improvement on the propagation algorithm; more substantial modifications of the propagation algorithm are discussed by Levoy.[3]

Tint fill is a significant improvement over interior fill for many soft-edged regions. However, it has two significant disadvantages:

1) The intensity at the old pixel is retained. This is *only* appropriate when the intensity of the new foreground is the same as that of the old foreground. For most renderings, this is not the case.
2) The algorithm only works against a black background.[1]

We wish to stress that these disadvantages apply only for RGB frame buffers; tint fill is "tuned" for a colour-mapped environment, and within that environment works for any region. Advances in technology have made full RGB frame buffers more viable, prompting our attempt to further expand the power of filling algorithms in the RGB environment.

At the end of his initial article, Smith[5] briefly presents a number of speculations about his algorithm which, as we understand them, could mitigate both of these disadvantages by using a number of special cases. Smith has since stated[7] that these speculations were not implemented; therefore, we will not consider these alternatives. Smith later solved the problem of filling against a single-coloured background,[6] but has not published or discussed the algorithm and therefore it cannot be evaluated here.

## 3. The Family of New Algorithms

We present a family of three new filling algorithms, each of which makes three assumptions:

(1) The background colour(s), whatever they may be, are known. While this is much less restrictive than assuming a black background, it may not hold for all applications. Furthermore, we stress that a "background colour" is any colour that contributes to the colour of the pixels within the region. It may be the case that no pixel purely consists of a background colour, and a background colour may not even correspond to a *physical* background region. For example, if the region is partially in shadow, black will be one of the background colours, even though there may be no black pixel, and no black "region" per se. Regions painted with fuzzy brushes and regions with specular highlights are similar examples.

(2) The anti-aliasing process operates on the linear space between colours; in other words, an anti-aliased pixel lying between $\beta$ colours will possess a colour that is a convex combination of those colours in colour space. This assumption is the key to all the algorithms; it holds for all additive colour spaces (XYZ, RGB, YIQ, etc.) due to the additive nature of light.

(3) No pixel is considered INSIDE twice. While this assumption is not true in Smith's original propagation algorithm,[5] it holds in his later algorithm,[6] and also holds for Levoy's.[3] If this assumption does not hold, it can be enforced by reserving one bit plane of the frame buffer to manually keep track of which pixels have been read.

The new algorithms are independent of propagation method; they can be implemented using either Smith's[5,6] or Levoy's[3] propagation methods.

Fundamentally, the three new algorithms presented solve a system of three equations in one, two, and three unknowns, respectively. As the number of unknowns increases, so does the power of the algorithm, as well as the computational expense. The algorithms approach the problem uniformly, and differ only in their generality and speed.

A note on notation: a colour will be represented as the vector triple $\mathbf{C}$. The components will be referred to as $C^{[0]}, C^{[1]}$, and $C^{[2]}$. In the case of a colour in the RGB system, $C^{[0]}, C^{[1]}$, and $C^{[2]}$ refer to the red, green, and blue components, respectively. In general, we use the notation of Table 3.1.

*Table 3.1: Notation and Symbols*

| Symbol | Meaning |
|---|---|
| $\mathbf{B}$ | colour of the background |
| $B^{[j]}$ | j'th component of the background colour |
| $\mathbf{B}_i$ | i'th background colour |
| $\mathbf{B}_0$ | old foreground colour ( $\equiv \mathbf{F}$) |
| $B_i^{[j]}$ | j'th component, i'th background colour |
| $\beta$ | number of background colours |
| $d$ | denominator/determinant |
| $\mathbf{F}$ | old foreground colour ( $\equiv \mathbf{B}_0$) |
| $\mathbf{G}$ | new foreground colour |
| $n$ | number of bits in each colour component |
| $N$ | number of values for each colour component |
| $\mathbf{P}$ | colour of an arbitrary pixel |
| $t$ | an interpolation factor |
| $\mathbf{T}$ | a vector of interpolation factors |
| $v_X$ | value of the colour $X$ |

### 3.1. Linear Fill

This section presents a simple, fast algorithm to soft fill against any single-coloured background, termed *linear fill.*

Given the assumptions of the previous section, the algorithm is almost embarrassingly simple. Let the colour of the old foreground be $\mathbf{F}$ (known by user specification), the background be $\mathbf{B}$ (known by assumption (1)), and the present pixel be $\mathbf{P}$. Then, by assumption (2), $\mathbf{P}$ is a linear interpolation between $\mathbf{F}$ and $\mathbf{B}$:

$$\mathbf{P} = t\,\mathbf{F} + (1-t)\,\mathbf{B}, \qquad (3.1)$$

where $0 \le t \le 1$ is the proportion of foreground colour in the present pixel colour.

The value of $t$ is the same across all components of the vector, and can be found from any component $i$ in which $F^{[i]} \neq B^{[i]}$. If more than one such component exists, the one with the largest value of $|F^{[i]} - B^{[i]}|$ is chosen to reduce discretization.[1] If there does not exist such an $i$, the algorithm will fail. This is a pathological case, in which the foreground and background are of the same colour. This case can be trivially detected in advance, and it is usually assumed that it will not occur.

The START procedure can be written as

find the dimension $i$ in which $|F^{[i]} - B^{[i]}|$ is largest.
$d = F^{[i]} - B^{[i]}$

The INSIDE procedure is

set $t$ to $(P^{[i]} - B^{[i]})/d$
if $t > 0$, pixel is inside the foreground with percentage $t$

And the SET procedure is

$\mathbf{P} = t\,\mathbf{G} + (1-t)\,\mathbf{B}$,
where $\mathbf{G}$ is the new foreground colour.

#### 3.1.1. Advantages of Linear Fill

Linear fill can be shown[1] to possess three significant advantages:

1) Linear fill is a proper generalization of tint fill

2) For an RGB frame buffer, linear fill is faster than tint fill, due to the expense of color system conversion.

3) Linear fill works for any single-coloured background.

#### 3.1.2. Limitations of Linear Fill

The linear fill algorithm fails in the pathological case when the foreground and background possess the same colour. Linear fill has a more significant limitation; even though it can fill against an arbitrary *single-coloured* background, it is not applicable if the background is composed of more than one colour, as is often the case in a complex shaded scene.

Formally, let the foreground colour be referred to as $\mathbf{F}$, as before, and the unique background colours as the set $\{ \mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_\beta \}$. Linear fill works if and only if $\beta = 1$. The most general filling algorithm should work for arbitrary $\beta$.

#### 3.1.3. Increasing $\beta$

Writing the components of equation (3.1) yields

$$P^{[0]} = tF^{[0]} + (1-t)B^{[0]},$$
$$P^{[1]} = tF^{[1]} + (1-t)B^{[1]},$$
$$P^{[2]} = tF^{[2]} + (1-t)B^{[2]},$$
$$\text{where} \quad 0 \le t \le 1$$

This is a system of three equations in one unknown. The next two algorithms, *triangle fill* and *tetrahedron fill* will replace this system of equations by similar systems in two and three unknowns. A fill algorithm in $\beta$ unknowns works for a set of background colours $\{ \mathbf{B}_1, \mathbf{B}_2, \cdots \mathbf{B}_\beta \}$. Unlike earlier algorithms for multi-coloured background filling,[2] we allow a pixel to have *any or all* $\beta$ background colours present in its composition. This is often the case for pixels at the intersection of regions, in shadow, in areas of specular reflection, etc.

### 3.2. Triangle Fill

Linear fill assumed a background composed of exactly one underlying colour, not equal to the foreground colour. Every pixel found lay on the line in colour space between the two colours. Triangle fill solves a slightly more difficult case, a background composed of exactly two underlying colours. In colour space, the background and foreground colours now form the vertices of a triangle; by assumption (2) of section (3), all pixels found will lie inside this triangle.

Let the colour of the foreground be $\mathbf{F}$, and the colour of an arbitrary pixel be $\mathbf{P}$, as before. Let the colours of the background be $\mathbf{B}_1$ and $\mathbf{B}_2$. Let $\mathbf{B}_0 = \mathbf{F}$ for notational convenience. Extending assumption (2) of linear fill to this problem, it follows that

$$\mathbf{P} = T^{[0]}\mathbf{B}_0 + T^{[1]}\mathbf{B}_1 + T^{[2]}\mathbf{B}_2.$$

This planar equation in three variables can be written as an equation in two variables, since $T^{[0]} + T^{[1]} + T^{[2]} = 1$.

$$\mathbf{P} = T^{[0]}\mathbf{B}_0 + T^{[1]}\mathbf{B}_1 + (1 - T^{[0]} - T^{[1]})\mathbf{B}_2$$

In terms of the unknowns,

$$\mathbf{P} - \mathbf{B}_2 = T^{[0]}(\mathbf{B}_0 - \mathbf{B}_2) + T^{[1]}(\mathbf{B}_1 - \mathbf{B}_2)$$

Writing in component form,

$$P^{[j]} - B_2^{[j]} = T^{[0]}(B_0^{[j]} - B_2^{[j]}) + T^{[1]}(B_1^{[j]} - B_2^{[j]}),$$
$$\text{for } j = 0,1,2$$

This is an over-constrained system of three equations in two unknowns, and can by solved by using Cramer's rule on

some two of the three equations:

$$T^{[0]} = \frac{\begin{vmatrix} P^{[i]}-B_2^{[i]} & B_1^{[i]}-B_2^{[i]} \\ P^{[j]}-B_2^{[j]} & B_1^{[j]}-B_2^{[j]} \end{vmatrix}}{d} \qquad (3.2)$$

$$T^{[1]} = \frac{\begin{vmatrix} B_0^{[i]}-B_2^{[i]} & P^{[i]}-B_2^{[i]} \\ B_0^{[j]}-B_2^{[j]} & P^{[j]}-B_2^{[j]} \end{vmatrix}}{d}$$

$$where\ i \neq j \in \{ 0,1,2 \},$$

$$\text{and } d = \begin{vmatrix} B_0^{[i]}-B_2^{[i]} & B_1^{[i]}-B_2^{[i]} \\ B_0^{[j]}-B_2^{[j]} & B_1^{[j]}-B_2^{[j]} \end{vmatrix}$$

### 3.2.1. Implementation

For triangle fill, the START procedure is:

$i \neq j \in \{ 0,1,2 \}$

find $i,j$ such that $d$ is maximized*,

The INSIDE procedure becomes:

Compute $T^{[0]}, T^{[1]}$ by equation (3.2).

If $T^{[0]}>0$, the pixel is inside the foreground with percentage $T^{[0]}$.

($T^{[0]}$ known in [0...1] by assumption (2) of section (3))

The SET procedure replaces the percentage of the old foreground with an equal percentage of the new, as in linear fill:

$$\mathbf{P} = T^{[0]}\mathbf{G} + T^{[1]}\mathbf{B}_1 + (1-T^{[0]}-T^{[1]})\mathbf{B}_2$$

### 3.2.2. Limitations of Triangle Fill

The START procedure of triangle fill must find the denominator of equation (3.2). Triangle fill will fail when the denominator is equal to zero for all choices of $i$ and $j$, $i \neq j$. Under what conditions will this hold? Assume the denominator *is* zero. Then

$$0 = (B_0^{[j]}-B_2^{[j]})(B_1^{[j]}-B_2^{[j]}) - (B_0^{[i]}-B_2^{[i]})(B_1^{[i]}-B_2^{[i]}),$$
$$\forall\ i,j \in \{ 0,1,2 \}, i \neq j$$

This set of scalar equations is exactly equivalent to the vector equation

$$\mathbf{0} = (\mathbf{B}_0-\mathbf{B}_2) \times (\mathbf{B}_1-\mathbf{B}_2),$$

This equation holds if and only if $(\mathbf{B}_0-\mathbf{B}_2)$ and $(\mathbf{B}_1-\mathbf{B}_2)$ are linear multiples of each other. This condition is true if and only if $\mathbf{B}_0$, $\mathbf{B}_1$, and $\mathbf{B}_2$ *are collinear*. If one of the colours is black, this condition arises if and only if the remaining two colours possess the same chromaticity (hue and saturation).

* As in linear fill, choosing the maximum denominator minimizes discretization

### 3.2.3. Triangle fill summary

To reiterate, triangle fill fills a region surrounded by two background colours. It is more expensive computationally than linear fill. The start-up cost of computing a 2-by-2 determinant is added, and the per-pixel processing is also increased. Unlike linear fill, which works whenever the foreground and background colours are not coincident, triangle fill requires a slightly stronger condition, that the background colours are non-collinear.

## 3.3. Tetrahedron fill

Tetrahedron fill uses the full power of the mathematics available, solving a system of three equations in three unknowns to fill against three background colours (the case $\beta$=3).

We wish to find $T^{[0]}, T^{[1]}, T^{[2]}, T^{[3]}$ such that

$$\mathbf{P} = T^{[0]}\mathbf{B}_0 + T^{[1]}\mathbf{B}_1 + T^{[2]}\mathbf{B}_2 + T^{[3]}\mathbf{B}_3 \qquad (3.3)$$

By fundamental linear algebra, three linearly independent three-dimensional colour vectors span the three-dimensional vector space. Both linear fill and triangle fill found a unique combination of fewer spanning vectors in smaller subspaces.

It may appear surprising that tetrahedron fill can find a solution, trying to find a unique linear combination of *four* vectors in a *three*-dimensional space. One extra degree of information, the knowledge that the sum of the weights is unity, allows this. Using this observation to set $T^{[0]} + T^{[1]} + T^{[2]} + T^{[3]} = 1$, equation (3.3) becomes an equation in three variables,

$$\mathbf{P} = T^{[0]}\mathbf{B}_0 + T^{[1]}\mathbf{B}_1 + T^{[2]}\mathbf{B}_2 + (1-T^{[0]}-T^{[1]}-T^{[2]})\mathbf{B}_3$$

In terms of the unknowns $T^{[0]}, T^{[1]}$, and $T^{[2]}$:

$$\mathbf{P}-\mathbf{B}_3 = T^{[0]}(\mathbf{B}_0-\mathbf{B}_3) + T^{[1]}(\mathbf{B}_1-\mathbf{B}_3) + T^{[2]}(\mathbf{B}_2-\mathbf{B}_3).$$

Let the row vector $\mathbf{T}$ be $\left[T^{[0]}, T^{[1]}, T^{[2]}\right]$. Then this can be rewritten as

$$\mathbf{P}-\mathbf{B}_3 = \mathbf{T} * \begin{bmatrix} \mathbf{B}_0-\mathbf{B}_3 \\ \mathbf{B}_1-\mathbf{B}_3 \\ \mathbf{B}_2-\mathbf{B}_3 \end{bmatrix}$$

This is one vector-valued equation in one vector-valued unknown ($\mathbf{T}$):

$$\left[\mathbf{P}-\mathbf{B}_3\right] = \mathbf{T}\overline{\mathbf{M}},\ where$$

$$\overline{\mathbf{M}} = \begin{bmatrix} \mathbf{B}_0-\mathbf{B}_3 \\ \mathbf{B}_1-\mathbf{B}_3 \\ \mathbf{B}_2-\mathbf{B}_3 \end{bmatrix} \qquad (3.4)$$

The unknown $\mathbf{T}$ can be found by matrix inversion or Gaussian reduction.

### 3.3.1. Tetrahedron fill procedures

The START procedure for tetrahedron fill inverts/reduces $\overline{\mathbf{M}}$. We denote the inverted $\overline{\mathbf{M}}$ by $\overline{\mathbf{M}}^{-1}$, the reduced $\overline{\mathbf{M}}$ by $\overline{\mathbf{U}}$.

The INSIDE procedure for tetrahedron fill is

$$\mathbf{T} = [\mathbf{P}-\mathbf{B}_3]\overline{\mathbf{M}}^{-1} \text{ for inversion}$$

$\mathbf{T}\overline{\mathbf{U}} = [\mathbf{P}\text{-}\mathbf{B}_3]$ for reduction
If $T^{[0]}>0$ , the pixel is inside with percentage $T^{[0]}$

The SET procedure becomes a trilinear interpolation:
$$\mathbf{P}=T^{[0]}\mathbf{G} + T^{[1]}\mathbf{B}_1 + T^{[2]}\mathbf{B}_2 + (1-T^{[0]}-T^{[1]}-T^{[2]})\mathbf{B}_3$$

### 3.3.2. Summary of tetrahedron fill

To reiterate, tetrahedron fill will fill a region surrounded by three background colours. It is more expensive computationally than triangle fill; a start-up cost of computing a matrix inverse is added, and the per-pixel processing is also increased. Tetrahedron fill is also not perfectly robust, requiring that the colours are not coplanar.

### 3.4. Power comparison

Each of the fill algorithms fails under certain conditions. We now pose two questions. First, if a fill against $\beta$ background colours fails, could the fill succeed by adding a dummy $(\beta+1)$'st background colour, and using a different algorithm? Second, if a fill against $\beta$ background colours succeeds, is it guaranteed that filling against any or all $\beta$–1 of the same background colours will succeed? We answer the first question in the negative with theorems 1 and 2, and the second in the positive with theorems 3 and 4. The proofs of the theorems can be found elsewhere.[1]

**Theorem 1**: *Linear failure implies triangle failure.*

**Theorem 2**: *Triangle failure implies tetrahedron failure.*

**Theorem 3**: *Triangle success implies linear success.*

**Theorem 4**: *Tetrahedron success implies triangle success.*

### 3.5. Summary of the three fills

A brief summary of the algorithms' characteristics is shown by Table 3.2.

*Table 3.2: Summary of fill algorithm expense*

| Procedure | Algorithm | | |
|---|---|---|---|
| | linear | triangle | tetrahedron' |
| START | minimal | find 2-by-2 determinant | invert 3-by-3 matrix |
| INSIDE | multiply | compute 2-by-2 determinant | multiply row vector by 3-by-3 matrix |
| SET | linear interpolation | bilinear interpolation | trilinear interpolation |
| fails when | $\mathbf{B}_0,\mathbf{B}_1$ coincident | $\mathbf{B}_0,\mathbf{B}_1,\mathbf{B}_2$ collinear | $\mathbf{B}_0,\mathbf{B}_1,\mathbf{B}_2,\mathbf{B}_3$ coplanar |

' using the inversion approach for tetrahedron fill.

An exact performance count has also been performed for the INSIDE and SET procedures in Table 3.3; the START expense is considered negligible. One minor optimization has been employed for the INSIDE procedures of triangle and tetrahedron fill; if $T^{[0]} = 0$, the procedure will not compute the other elements of **T**.

Assuming that addition and subtraction, and multiplication and division are each approximately equivalent in time, the computational expense monotonically increases among algorithms.

*Table 3.3: Performance comparison for a region with* i *interior and* b *boundary pixels.*

| Operation | Algorithm | | |
|---|---|---|---|
| | Linear | Triangle | Tetrahedron' |
| tests | i+ b | i+ b | i+ b |
| := | 4i + b | 5i+ b | 6i+ b |
| + | 3i | 6i | 15i+ 2b |
| - | 4i + b | 16i+ 5b | 12i+ 3b |
| * | 6i | 13i+ 2b | 21i+ 3b |
| / | i + b | 2i+ b | 0 |
| reads | i+ b | i+ b | i+ b |
| writes | i | i | i |

using the inversion approach for tetrahedron fill.

## 4. Conclusion

This paper has presented a family of new algorithms for soft filling. The computational expense and power increase monotonically between the algorithms. The specific algorithm that is "best" is determined solely by the particular rendering: linear fill when only one background colour exists, triangle fill for two background colours, and tetrahedron fill for three.

There remain two significant problems in the field: finding the background colours without user help, and filling a region with more than three background colours.

### Acknowledgements

The authors wish to thank Marc Levoy of Hanna-Barbera and Tony DeRose, Paul Hilfinger, and Rick Speer of Berkeley for their help in this project.

### References

1. Kenneth P. Fishkin and Brian A. Barsky, "A Family of New Algorithms for Soft Filling," in *Proceedings of SIGGRAPH '84*, Minneapolis(July 23-27, 1984). Accepted for publication.
2. Marc S. Levoy, *Computer-Assisted Cartoon Animation,* Master's Thesis, Cornell University, Ithaca, N.Y. (August, 1978).
3. Marc S. Levoy, *Area Flooding Algorithms,* Report, Hanna-Barbera Productions (June, 1981).
4. Henry Lieberman, "How To Color in a Coloring Book," pp. 111-116 in *Proceedings of SIGGRAPH '78*, ACM,(1978).
5. Alvy Ray Smith, "Tint Fill," pp. 276-283 in *Proceedings of SIGGRAPH '79*, ACM,(August, 1979). Also Technical Memo No. 6, New York Institute of Technology.
6. Alvy Ray Smith, *Fill Tutorial Notes,* Report No. 40, LucasFilm (1981).
7. Alvy Ray Smith, private communication. September, 1982.