
Eindhoven, augustus 2000

De faculteit Elektrotechniek van de Technische Universiteit Eindhoven aanvaardt geen aansprakelijkheid voor de inhoud van dit verslag.
Edge Anti-aliased
Two Pass Forward Texture Mapping

Author: A.van Loenen (397909)
Supervisors: ing. K. Meinds (Nat.Lab Eindhoven)
dr.ir.C.W.A.M van Overveld (Fac W&I)
prof.dr.dipl.ing. D.K Hammer (Fac W&I)
prof.dr.A.A.C.M.Kalker (Fac E)
Date: 22-08-2000
Location: Philips Nat.Lab Eindhoven
Abstract

This document presents the results of my graduation project carried out at Philips Nat.Lab Eindhoven. The research project focused on extending to a two pass forward texture mapping implementation by K.Meinds [12]. This extension makes it possible to generate high quality edge anti-aliased images of texture mapped polygons. Without requiring extensive modification to the existing two pass forward texture mapping implementation by K.meinds it is possible to generate correctly anti-aliased images for texture minification situations. It is more difficult to generate edge-anti aliased images in case of texture magnification. In this
Acknowledgements

I would like to thank Wilma and Scooby for their excellent job at supporting me throughout my graduation project. Furthermore I would like to thank my mother who has supported me in the first few years of my course at the TUE. Also, a big thank you to SRI of which the members frequently reminded me of the humourus side of life which kept me in good spirit. Last but not least I want to thank ing.K.Meinds (Philips Nat.Lab Eindhoven), dr.ir.C.W.A.M van Overveld (TUE), prof.dr.A.A.C.M.Kalker (TUE) and prof.dr.dipl.ing. D.K Hammer for making my graduation project possible.
1 Introduction

Traditional high quality texture mapping requires computationally expensive 2D filtering. This can be avoided by using two pass forward texture mapping [4] (Figure 1a) which uses computationally less expensive 1D filter operations. Another advantage is that texture memory access is more efficient due to the regular sequential access pattern. An implementation of two pass forward texture mapping has been researched by K.Meinds at Philips Nat.Lab Eindhoven [12]. This implementation did however not generate edge anti-aliased images. Edge anti-aliasing suppresses the phenomenon of jagged edges (Figure 1b) which can result in disturbing visual artefacts in an animated sequence of a texture mapped object. A discussion on the cause of aliasing problems can be found in [7]. This document presents an extension to the initial two pass forward texture mapping implementation by K.Meinds which is capable of generating high quality edge anti-aliased images of texture mapped polygons (Figure 1c).

![Figure 1](image)

Figure 1. (a) two pass texture mapping of a triangle, the bottom right picture represents the final texture mapped triangle as it would appear on the screen. (b) close-up aliased edge. (c) close-up anti-aliased edge.
2 Two-Pass Forward Texture Mapping Introduction

The forward warp of a continuous texture signal $I_c(u, v)$ in texture space onto a continuous projection window in $(x, y)$ screen space resulting in $I_c,s(x, y)$ can be described by:

$$I_c(u, v) = I_c,s\left(F_x(u, v), F_y(u, v)\right) = I_c,s(x, y)$$ (1)

Where:

$$F_x(u, v) = \frac{Au + Bv + C}{Gu + Hv + I}$$ (2)

and

$$F_y(u, v) = \frac{Du + Ev + F}{Gu + Hv + I}$$ (3)

For an analysis of inferring the relationship see [2]. This forward projection can also be computed in two passes which was first described by Catmull, E and A.R. Smith [4]. Two pass forward texture mapping generates the final image as it should appear on the projection window in two passes, warping the image along an axis in each pass (Figure 2). In the first pass, the texture is warped horizontally according to:

$$I_c(u, v) = I_c,s\left(F_x(u, v), v\right) = I_c,s(x, v)$$ (4)

This results in an intermediate image $I_c,s(x, v)$ in intermediate space which is a horizontally sheared version of the input image. In general, the shear is not constant resulting in a curved intermediate image.

![Figure 2, Two pass forward warp](image-url)
In the second pass, the intermediate image is warped vertically according to:

\[ I_{c, v}(x, v) = I_{c, x}(x, F_y(H_u(x, v), v)) = I_{c, x}(x, y) \]  

(5)

Where \( H_u(x, v) \) is an auxiliary function computing \( u \) as function of \( x \) and \( v \). This function can be derived by rearrangement of (2). Substitution of \( H_u(x, v) \) into (3) allows the vertical warp function to be expressed directly as a function of \( x \) and \( v \):

\[ y = F_y(x, v) = \frac{Rx + T - Nx - Q}{-S + P} \]  

(6)

The short hand notation \( I(u, v) = I(x, v) \), and similarly \( I(x, v) = I(x, y) \) will be used from now specifying the relation that \( I(u, v) \) maps to \( I(x, v) \) and \( I(x, v) \) maps to \( I(x, y) \) as determined by the warp formulas (2) and (6), where it is understood that \( I(u, v) \) represents the signal in texture space, \( I(x, v) \) the signal in intermediate space and \( I(x, y) \) the signal in screen space.

The texture, the intermediate space and the projection window are all discrete in a discrete implementation of two pass forward texture mapping. In order to obtain a discrete image on the discrete projection window representing a high quality approximation of the continuous texture on the continuous projection window, quality filtering should be used. This chapter will continue by first introducing the reader to a general forward texture mapping model incorporating high quality filtering. This model is presented to familiarize the reader with forward texture mapping in general, after which two pass forward texture mapping using quality filtering will be discussed. The general forward texture mapping model also serves as a high level model for analysis with regards to researching an edge anti-aliasing scheme. This allows a two pass forward texture mapping based edge anti-aliasing scheme to be derived afterwards.
2.1 Forward Texture Mapping Model

The analysis presented here presents a summary of [11]. The forward texture mapping model is described by the following steps:

- **Reconstruction**: Reconstruct the discrete texture signal back into its continuous representation.
- **Warping**: Warp the continuous texture signal to screen space.
- **Prefiltering**: Bandlimit the texture signal with respect to the sample grid in screen space.
- **Sampling**: Sample the texture signal at the sample positions in screen space.

The discrete texture is assumed to be obtained by point sampling a continuous texture signal \( I_{\mu,v} \) which has occurred somewhere in the past \(^1\):

\[
I_{\mu,v} = I(u,v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I_c(m\Delta u, n\Delta v) \delta(u - m\Delta u, v - n\Delta v) \tag{7}
\]

Where:

- \( \Delta u, \Delta v \): The sampling distances of the discrete texture.

With no loss of generality, \( \Delta u \) and \( \Delta v \) can be assumed to be unity. A sample of the discrete texture in texture space will be referred to as a texel. Since \( \Delta u, \Delta v \) are assumed to be unity, the texel positions take on integer values. A texel position will be denoted as \((\overline{u}, \overline{v}) \in \mathbb{N}^2\). The steps describing the model are discussed next.

2.1.1 Reconstruction

The reconstruction operation reconstructs the discrete texture \( I_{\mu,v} \) resulting in the continuous reconstructed texture \( I_r(u,v) \). This is done by weighing the intensities of the discrete samples with a 2D reconstruction filter function:

\[
I_r(u,v) = \sum_{(\overline{u}, \overline{v}) \in A_r(u,v)} h_r((u,v) - (\overline{u}, \overline{v})) \cdot I_{\mu,v} (\overline{u}, \overline{v}) \tag{8}
\]

Where:

- \( h_r(u,v) \): The 2D reconstruction filter function
- \( A_r(a,b) \): The collection of \((u,v)\) positions \( \in \mathbb{R}^2 \) inside the 2D reconstruction filter footprint centered around \((a,b)\).

2.1.2 Warping

The reconstructed signal is warped according to \( I_r(u,v) = I_r(x,y) \).

\(^1\) \( I_{\mu,v} \) is expected to be bandlimited to the Nyquist frequency of the discrete texture grid. The Nyquist frequency is the maximum allowed frequency in a signal for it to be correctly resampled to a resample grid without introducing aliasing. In this case, the Nyquist frequency is equal to half the sampling frequency of the discrete texture grid.
2.1.3 Prefiltering

After the reconstructed signal is projected onto the discrete projection window, the signal is prefiltered to ensure that the signal is bandlimited to the Nyquist frequency of the projection sample grid. The prefiltered signal on the projection window is given by $I_p(x, y)$:

$$I_p(x, y) = \int_{(x', y') \in A_p(x, y)} h_p((x, y) - (x', y')) \cdot I_c(x', y') \cdot d(x', y')$$

(9)

Where:
- $h_p(x, y)$ : The 2D prefilter function.
- $A_p(a, b)$ : The collection of $(x, y)$ positions $\in \mathbb{R}^2$ inside the 2D prefilter footprint centered around $(a, b)$

2.1.4 Sampling

The final discrete image $I(x, y)$ on the discrete projection window is given by sampling of $I_p(x, y)$:

$$I(x, y) = \sum_{m=0}^{W} \sum_{n=0}^{H} I_p(m\Delta x, n\Delta y) \delta(x - m\Delta x, y - n\Delta y)$$

(10)

Where:
- $W/H$ : The maximum horizontal/vertical projection window coordinate
- $\Delta x, \Delta y$ : The sampling distances of the discrete projection window grid.

With no loss of generality, $\Delta x$ and $\Delta y$ will be assumed to be unity. A sample in screen space on the discrete projection window grid will be referred to as a pixel. Since $\Delta x, \Delta y$ are assumed to be unity, a pixel position takes on integer values. A pixel position will be denoted as $(\bar{x}, \bar{y}) \in \mathbb{N}^2$. Since sampling only occurs at pixel positions, without loss of generality the formula for the intensity at a pixel $(\bar{x}, \bar{y})$ can be simplified to:

$$I(\bar{x}, \bar{y}) = \int_{(x, y) \in A_p(\bar{x}, \bar{y})} h_p((\bar{x}, \bar{y}) - (x, y)) \cdot I_c(x, y) \cdot d(x, y)$$

(11)
2.2 Two Pass Forward Texture Mapping Model

The two pass forward texture mapping equations can now be derived from the forward texture mapping model presented in 2.1.

2.2.1 Formula derivations

Recall that in the high quality forward texture mapping model, the intensity at a pixel $(x, y)$ is given by:

$$ I(x, y) = \frac{\int_{(x, y) \in A_p(x, y)} h_p((x, y) - (x, y)) \cdot I_r(x, y) \cdot d(x, y)}{\int_{(x, y) \in A_p(x, y)} d(x, y)} \quad (12) $$

(12) can be rewritten into a separable form:

$$ I(x, y) = \frac{\int_{y \in A_{p,y}^{(x)}} h_{p,y}(y - y) \cdot \int_{x \in A_{p,x}^{(y)}} h_{p,x}(x - x) \cdot I_{r,sep}(x, y) \quad dx \quad dy}{\int_{y \in A_{p,y}^{(x)}} \int_{x \in A_{p,x}^{(y)}} d(x, y)} $$

$$ I_{r,sep}(u, v) = \sum_{v \in A_{r,v}^{(u)}} h_{r,v}(v - v) \cdot I_{r,sep}(u - u) \cdot I_d(u, v) \quad (13) $$

Where:

- $h_{p,x/y}(x/y)$: The 1D horizontal/vertical prefilter function.
- $A_{p,x/y}(a)$: The collection of $x/y$ positions $\in \mathbb{R}$ inside the 1D prefilter footprint centered around $a$.
- $h_{r,x/y}(u/v)$: The 1D horizontal/vertical reconstruction filter function
- $A_{r,x/y}(a)$: The collection of $u/v$ positions $\in \mathbb{R}$ inside the 1D reconstruction filter footprint centered around $a$.

And $I_{r,sep}(u,v) = I_{r,sep}(x,y)$.

Furthermore it is required that the 2D filters are separable, requiring $h_p(x,y) = h_{p,x}(x) \cdot h_{p,y}(y)$ and $h_r(u,v) = h_{r,u}(u) \cdot h_{r,v}(v)$. (13) and (14) in turn can be reordered again resulting in the formulas for horizontal and vertical pass of the two pass approach:

The intensity of an intermediate sample $(\bar{x}, \bar{y})$ is computed in the horizontal pass and is determined by:

$$ I_{int}(x, y) = \frac{\int_{y \in A_{p,y}^{(x)}} h_{p,y}(y - y) \cdot I_{r,h}(x, y) \quad dx}{\int_{y \in A_{p,y}^{(x)}} \int_{x \in A_{p,x}^{(y)}} d(x, y)} \quad (15) $$

Where $I_{r,h}(u, \bar{v}) = I_{r,h}(x, \bar{y}), I_{r,sep}(u, \bar{v})$ being determined by:

$$ I_{r,h}(u, \bar{v}) = \sum_{\bar{u} \in A_{r,v}^{(u)}} h_{r,u}(u - u) \cdot I_d(u, \bar{v}) \quad (16) $$

In order for the separable filtering to be correct, these formulas actually only hold when the $u$-axis is aligned with the $x$-axis, and the $v$-axis with the $y$-axis. However, the formulas will be used for all projections of $(u,v)$ space onto $(x,y)$ space, the error simply being accepted as a consequence of using the two pass approach.
The intensity for a pixel \((x, y)\) is computed in the vertical pass and is determined by:

\[
I(x, y) = \int_{y \in A_{x, y}(y)} h_{p, y}(y - y) \cdot I_{r,v}(x, y) \cdot dy
\]

(17)

Where \(I_{r,v}(x, v) = I_{r,v}(x, v)\) being determined by:

\[
I_{r,v}(x, v) = \sum_{y \in A_{x,v}(v)} h_{r,v}(v - v) \cdot I_{int}(x, v) \cdot dy
\]

(18)

In computing the intermediate sample intensities the following model is assumed:

**Horizontal pass (for each texture row in texture space):**
- Reconstruct the texture signal horizontally using a 1D horizontal reconstruction filter
- Warp the signal horizontally onto the corresponding row in intermediate space
- Obtain the intensities of the intermediate samples by applying a 1D horizontal prefilter at each intermediate sample position

An example for the horizontal pass is shown in Figure 3. For the row \(v = \tilde{v}\), the signal is reconstructed using the texels. This signal is horizontally warped onto the intermediate row. An intermediate sample position with its corresponding prefilter is shown. The intensity of the intermediate sample is obtained by applying the prefilter on the signal in the prefilter footprint (red).
In computing the pixel intensities the following model is assumed:

**Vertical pass (for each intermediate column in intermediate space):**
- Reconstruct the intermediate signal vertically using a 1D vertical reconstruction filter
- Warp the signal vertically onto the corresponding column in screen space
- Obtain the intensities at the pixels by applying a 1D vertical prefilter at each pixel

An example for vertical pass is shown in Figure 4. For the column $x=x_i$, the signal is reconstructed using the intermediate sample intensities. This signal is vertically warped onto the pixel column. A pixel with its corresponding prefilter is shown. The intensity of the pixel is obtained by applying the prefilter on the signal in the prefilter footprint (red).

---

![Diagram](image.png)

**Figure 4.** Resampling a column in the vertical pass
3 Deriving an Edge Anti-Aliasing Forward Texture Mapping Model

It is now assumed that the discrete texture signal is to be mapped onto a convex polygon of which its vertices are known in texture space. It is desired that the texture mapping should result in an edge anti-aliased image. In this chapter, an edge anti-aliasing model will be derived for the forward texture mapping model presented in paragraph 2.1. The two pass forward texture mapping implementation will be discussed in chapter 4. Because it is desired to research incorporating edge anti-aliasing into the two pass forward texture mapping algorithm which resamples on a per-polygon basis, this thesis is restricted in the sense that it will only focus on anti-aliasing schemes which generate edge anti-aliasing information on a per-polygon basis. Consider now a pixel on the outside near a polygon edge on the projection window. For an edge anti-aliasing scheme for a polygon based resampling model it is required that the resampling of the discrete texture mapped onto the polygon should result in a partial intensity contribution for this pixel. A general formula for the contribution of a texture mapped polygon \( n \) to the intensity of a pixel \((x, y)\) is given by:

\[
I(x, y) = \sum_{\mathcal{V}(u, v)} \alpha_{n}(\mathcal{U}, \mathcal{V}) \cdot I_{d,n}(u, v)
\]

Note that (19) doesn’t necessarily assume a reconstruction & prefiltering model, but implements anti-aliasing in a general fashion by simply contributing a weighted sum of the texel intensities from the discrete texture for each polygon. The coefficients can be chosen in such a way that it allows for an efficient implementation while still generating a visually pleasing image. However, given the forward texture mapping model framework it is desired to describe an edge anti-aliasing scheme in terms of reconstruction and prefiltering (which implicitly would prescribe the coefficients in (19)). The edge anti-aliasing scheme for the forward texture mapping model will be based on the edge anti-aliasing scheme found in raytracing. Although raytracing in itself does not represent a polygon based resampling model, it does allow for a high quality edge anti-aliasing scheme to be derived for the forward texture mapping model. In ideal raytracing an intensity at a pixel is obtained as follows: An infinite amount of rays are shot from the projection point (eye) through its corresponding prefilter footprint (Figure 5). Each ray will intersect the prefilter footprint and optionally a texture mapped polygon in the scene. If it intersects the surface of a polygon, then signal reconstruction is performed at the intersection position \( \gamma \). This reconstructed intensity is projected to the corresponding intersection position in the prefilter footprint. Shooting the infinite amount of rays will result in a 2D continuous signal in the prefilter footprint. The pixel intensity is finally obtained by weighing the 2D signal in the prefilter footprint with the prefilter function.

---

2 If a ray doesn’t hit any polygon, the intensity is normally taken to be a predefined ambient value. Without loss of generality, the intensity is assumed to be zero.
The intensity of a pixel \((x, y)\) according to the raytracing model is determined by:

\[
I(x, y) = \sum_{n=1}^{N} \int_{(x, y) \in A_p(x, y)} h_p((x, y) - (x, y)) \cdot V_n(x, y) \cdot I_{r,n}(x, y) \cdot d(x, y)
\]

(20)

Where:
- \(N\): The number of polygons in the scene
- \(V_n(x, y)\): 1 if \((x, y)\) lies in the visible interior of polygon \(n\) on the projection window, 0 otherwise.
- \(I_{r,n}(x, y)\): The reconstructed signal on the projection window belonging to polygon \(n\).

The 2D signal in the prefilter footprint in general can be seen to be a composition of warped reconstructed signals originating from different polygons. Although this composite 2D signal will contain signal discontinuities at the projected edges of polygons, edge anti-aliasing is obtained in a natural way since the prefilter acts as a blend operator. The same result can be seen to be obtained by the forward texture mapping model as presented in paragraph 2.1 if the reconstructed signal for each polygon being resampled is bounded to its visible polygon interior:

The intensity contribution to the intensity of pixel \((x, y)\) due to resampling a given polygon \(n\) would then be determined by:

\[
I_n(x, y) = \int_{(x, y) \in A_p(x, y)} h_p((x, y) - (x, y)) \cdot V_n(x, y) \cdot I_{r,n}(x, y) \cdot d(x, y)
\]

(21)

The final intensity of a pixel \((x, y)\) would be then be determined by:

\[
I(x, y) = \sum_{n=1}^{N} I_n(x, y)
\]

(22)

Which can be seen to be equal to (20). The intensity contribution to the pixel \((x, y)\) resampling a given polygon \(n\) can be expressed as a fraction, which can be explicitly determined by:

\[
\lambda_n(x, y) = \int_{(x, y) \in A_p(x, y)} h_p((x, y) - (x, y)) \cdot V_n(x, y) \cdot d(x, y)
\]

(23)

This fraction will be referred to as the intensity blend fraction \(\lambda\), signifying the relative contribution of the intensity contribution to the intensity of a pixel. The intensity blend fractions sum up to one if the prefilter footprint of a pixel is totally overlapped by polygons:

\[
\sum_{n=1}^{N} \lambda_n(x, y) = 1
\]

(24)

Unfortunately (21) and (23) cannot be readily implemented due to a 3D graphics architecture restriction being imposed:
- Other polygons might overlap the current polygon being texture mapped as seen from the projection window. This might restrict its final visible interior region on the projection window. While resampling the current polygon, no information is known about which region of its interior will finally be visible.

Therefore the reconstructed signal of a polygon can only be bound against the polygon interior and not the visible polygon interior region, resulting in the following intensity contribution at a pixel \((x, y)\) due to resampling a given polygon \(n\):

\[
\lambda_n(x, y)
\]

Note that this is not to be confused with the coverage fraction for which the definition in this thesis is the fraction of the visible polygon area inside the prefilter footprint.
\[ I_n(x, y) = \left[ h_p((x, y) - (x, y)) \cdot B_n(x, y) \cdot I_r(x, y) \cdot d(x, y) \right] \]

Where:
- \( B_n(x, y) \): 1 if \((x,y)\) lies in the interior of polygon \( n \) on the projection window, 0 otherwise.

The blend fraction of the intensity contribution at pixel \((x, y)\) due to resampling a given polygon \( n \) therefore being determined by:

\[ \lambda_n(x, y) = \frac{h_p((x, y) - (x, y)) \cdot B_n(x, y) \cdot I_r(x, y) \cdot d(x, y)}{h_p((x, y) - (x, y)) \cdot B_n(x, y) \cdot I_r(x, y) \cdot d(x, y)} \]

It can be seen that the intensity contribution and blend fraction at a pixel are incorrectly computed if the interior region of the polygon within the prefilter footprint is unequal to the final visible region of the polygon within the prefilter footprint. An example of this can be seen in Figure 6, where the pixel prefilter footprint contains parts of both polygon A and B. Assuming polygon B overlaps polygon A, the intensity contribution and blend fraction of polygon A will be incorrectly computed. Note further that this cannot be corrected for by rearranging the polygon resampling order.

Figure 6 An incorrect intensity contribution and blend fraction will be computed for polygon A

Without the 3D graphics architecture restriction as imposed earlier, a trivial composition unit in the 3D graphics pipeline would merely need to add the intensities at each pixel. However, in this case a non-trivial composition unit is required. Such a composition unit ideally will remember all the intensity- and blend fractions at a pixel together with the polygon regions in the prefilter footprint, which it will use to rescale intensities after all polygons have been processed. Composition units will not be researched in this thesis. Examples of composition units can for example be found in [3] and [16]. However, the two pass forward mapping implementation is to generate information for such a composition unit. For this thesis, a low quality composition unit was allowed to be assumed. Therefore the two pass forward texture mapping implementation will only need to generate the intensity contributions and blend fractions at the pixels, but not its interior region.
4 Implementation

This chapter discusses the implementation details of the edge anti-aliased two pass forward texture mapping implementation, based on the edge anti-aliasing forward texture mapping model as presented in chapter 3.

4.1 Formulas

Three bounding operators are defined as follows:

- $B_{lu}(u,v): 1$ for interior polygon positions in texture space, 0 else where
- $B_{lr}(x,y): 1$ for interior polygon positions in screen space, 0 else where
- $B_{li}(x,v): 1$ for interior polygon positions in intermediate space, 0 else where

A horizontal polygon span on a texture row on $v = v_k$ is defined as:

$$[e_{lu}, e_{lu}] = \left\{ v \in \mathbb{R} : B_{lu}(u,v) = 1 \right\}$$  (27)

A horizontal polygon span on an intermediate row on $v = v_k$ is defined as:

$$[e_{li}, e_{li}] = \left\{ v \in \mathbb{R} : B_{li}(x,v) = 1 \right\}$$  (28)

A vertical polygon span on an intermediate column $x = x_k$ is defined as:

$$[e_{li}, e_{li}] = \left\{ x \in \mathbb{R} : B_{li}(x,v) = 1 \right\}$$  (29)

A vertical polygon span on a pixel column $x = x_k$ is defined as:

$$[e_{li}, e_{li}] = \left\{ y \in \mathbb{R} : B_{li}(x,y) = 1 \right\}$$  (30)

The formulas for the intermediate intensities and pixel intensities using the two pass forward texture mapping approach are derived from the forward texture mapping formula (25) similar as discussed in paragraph 2.2.1. The formulas become:

The intensity of an intermediate sample $(\bar{x}, \bar{v})$:

$$I_{int}(\bar{x}, \bar{v})_{v_k} = \left\{ h_{r,s}(\bar{x} - x) \cdot B_{lr}(\bar{x}, v_{r,s}) \cdot I_{r,s}(x, v) \cdot dx \right\}$$  (31)

Where $I_{r,s}(u, v) = I_{r}(x, v), I_{r,s}(u, v)$ being determined by:

$$I_{r,s}(u, v) = \sum_{\forall u \in \mathbb{A}_{r,s}(u)} h_{r,s}(u - u) \cdot I_{d}(u, v)$$  (32)

The intensity of a pixel $(\bar{x}, \bar{y})$:

$$I(\bar{x}, \bar{y}) = \left\{ h_{r,s}(\bar{y} - y) \cdot I_{r,s}(\bar{x}, y) \cdot dy \right\}$$  (33)
Where \( I_r, s(\bar{x}, v) = I_r, s(\bar{x}, y), I_r, s(\bar{x}, v) \) being determined by:

\[
I_r, s(\bar{x}, v) = \sum_{v \in A_{r, s}(v)} h_r, s(v - v) \cdot I_{\text{int}}(x, v)_{v_r,v}
\]  

(34)

These formulas can however not be readily implemented using a horizontal- and vertical pass since the intensities of the intermediate samples are dependent of the vertical reconstruction required for computing a pixel intensity. This is shown in (31) by the dependency of \( I_{\text{int}}(\bar{x}, \bar{v}) \) on \( v_r,v \), \( v_r,v \) representing a vertical reconstruction position. Consider Figure 7 where the intermediate resample grid is shown. Also, the position \( (\bar{x}, v_{r,l}) \) is shown for which the vertical reconstructed intensity is to be computed. Assuming the reconstruction filter requires 4 samples for reconstruction then reconstruction at \( (\bar{x}, v_{r,l}) \) requires the intensity values of the four intermediate samples shown (green dots). For each intermediate sample its intensity is computed by applying a horizontal prefilter on the signal being warped onto its intermediate resample row. For vertical reconstruction at \( (\bar{x}, v_{r,l}) \), each of these signals will be bounded by the horizontal polygon span \( \{e_0, e_1, h \} \) on \( v = v_{r,l} \). In Figure 7 the green lines mark the areas where the horizontal signals on the four intermediate sample rows are bounded to. These segments will be referred to as the signal reconstruction segments from now on, determining where the reconstructed signal exists.

![Figure 7 Vertical reconstruction at \( (\bar{x}, v_{r,l}) \)](image)

For the four intermediate samples shown, the bounded signal lying inside the horizontal prefilter footprints (grey areas) will be used to compute the prefiltered intensity. The problem is that in general, \( B_r(\bar{x}, v_{r,l}) \) will vary with \( v_{r,l} \). The signal reconstruction segments on the intermediate sample rows therefore depend on the position where vertical reconstruction takes place. This can be seen by comparing Figure 8 to Figure 7. In
Figure 8, the vertical reconstruction position \((\bar{x}, v_r)\) is shown. It can be seen that the signal reconstruction segment on the intermediate resample row of intermediate sample \((\bar{x}, v_r)\) is different. The intensity value of the intermediate sample \((\bar{x}, v_r)\) which is used in reconstruction at both \((\bar{x}, v_r)\) and \((\bar{x}, v_r')\) can therefore not be uniquely determined in a horizontal pass. In general, the intermediate sample intensities needed for vertical reconstruction would have to be recomputed for each vertical reconstruction position. Formula (31) which computes the intermediate sample intensities will therefore have to be approximated in order for the intermediate sample intensities to be computed independent from vertical reconstruction, allowing a horizontal- and vertical pass.

![Diagram](image)

**Figure 8, Vertical reconstruction at \((\bar{x}, v_r)\)**

In order to allow the computations of the intermediate samples to be independent of the computations required in the vertical pass, the following approximation is used for computing the intermediate sample intensities in the horizontal pass:

\[
I_{\text{int}}(\bar{x}, \bar{v}) = \int_{x \in A_{p_x}(\bar{x})} h_{p \times} (\bar{x} - x) \cdot B_i(x, \bar{v}) \cdot I_{r,h}(x, \bar{v}) \cdot dx
\]

(35)

In (35), the term \(B_i(x, \bar{v})\) is approximated by using the term \(B_i(x, \bar{v})\). According to this approach, the signal reconstruction segment on an intermediate row \(v = \bar{v}_j\) is determined by the horizontal polygon span on \(v = \bar{v}_j\) (Figure 9) and is therefore no longer dependent on the horizontal polygon span belonging to a vertical reconstruction position in the vertical pass. The intermediate samples can now be uniquely determined in the horizontal pass according to the model presented in paragraph 2.2 (4). The effect of using \(B_i(x, \bar{v})\) in

---

4 The actual implementation of the model presented 2.2 will be different as will be seen in 4.3
instead of using $B_i(x, v, \nu)$ can be seen by comparing Figure 9 to Figure 7. Figure 9 shows the same vertical reconstruction position $(\tilde{x}, v, \nu)$ together with its intermediate samples and signal reconstruction segments (green lines). The error made due to the approximation can be seen by comparing Figure 9 and Figure 7. In Figure 9 the horizontal prefilters of the two intermediate samples lying above the vertical reconstruction position will prefilter more signal information, resulting in higher intensity values for these intermediate samples. The horizontal prefilters intermediate samples lying below the vertical reconstruction position will prefilter less signal information, resulting in a lower intensity values for these samples.

![Diagram of signal reconstruction segments](image)

Figure 9, Fixed signal reconstruction segments for the intermediate samples

The relative intensity error of an intermediate sample $(\tilde{x}, v)$ used for vertical reconstruction in the vertical pass at a position $(\tilde{x}, v)$ due to the approximation can be determined by:

$$E_{I_{\nu}}(\tilde{x}, v, \nu) = \int_{x \in A_{\nu, \nu}(\tilde{x})} h_{\nu, \nu}(x - \tilde{x}) \cdot \{B_i(x, v) - B_j(x, v)\} \cdot dx$$  \hspace{1cm} (36)

The relative error of an intermediate sample used for reconstruction at $(\tilde{x}, v)$ is thus determined by the mismatch in the horizontal signal reconstruction segment area in its prefilter footprint. The mismatch is due to using the horizontal polygon span on its row for determination of the horizontal signal reconstruction segment, rather than using the horizontal polygon span on the vertical reconstruction position.

The relative error in the intensity value obtained by vertical reconstruction at a certain $(\tilde{x}, v)$ becoming:

$$E_{I_{\nu}}(\tilde{x}, v) = \sum_{\nu \in A_{\nu, \nu}(\tilde{x})} h_{\nu, \nu}(v - \nu) \cdot E_{I_{\nu}}(\tilde{x}, v, \nu)$$  \hspace{1cm} (37)
The relative error in the final intensity at a pixel \((x',y')\) becoming:

\[
E_I(x',y') = \int_{\mathcal{A}_{p_x}(y')} h_{f_x}(y-y') \cdot E_{I_x}(x',y') \cdot dy
\]  

(38)

Where \(E_{I_{x'}}(x',y) = E_{I_{x'}}(x,y)\)

### 4.1.1 Blend fraction formulas

It can be observed that the computation of the blend fraction (26) can be evaluated by (25) which computes the intensities, only requiring \(I_{x}(x,y)\) to be taken equal to 1. The two pass formulas for computing intensities given by (35) and (33) can therefore also be used to compute the blend fraction. An advantage can be gained from this, since it would allow the blend fraction to either be implemented using the same resampling hardware or to be computed by reusing the intensity resample hardware.

Using (35), the horizontal pass computes a horizontal 1D blend fraction at an intermediate sample \((\bar{x},\bar{y})\):

\[
\hat{\lambda}_{\text{int}}(\bar{x},\bar{y}) = \int_{\mathcal{A}_{p_x}(\bar{y})} h_{p_x}(\bar{x}-x) \cdot B_{i}(x,\bar{y}) \cdot 1 \cdot dx
\]  

(39)

And using (33), the vertical pass computes the final blend fraction at a pixel \((x',y)\):

\[
\hat{\lambda}(x',y) = \int_{\mathcal{A}_{p_y}(y')} h_{p_y}(y-y') \cdot \hat{\lambda}_{i_x}(x',y) \cdot dy
\]  

(40)

Where \(\hat{\lambda}_{i_x}(x',y) = \hat{\lambda}_{i_x}(x,y)\), \(\hat{\lambda}_{i}(x',y)\) being determined by:

\[
\hat{\lambda}_{i}(x',y) = \sum_{\mathcal{A}_{p_x}(y')} h_{i_x}(y-y') \cdot \hat{\lambda}_{\text{int}}(x',y')
\]  

(41)

Note that since (35) computes the intensities using an approximation, the horizontal blend fractions for the intermediate samples and the final blend fractions for the pixels are subjected to the same errors as the intensity computations given by (36) and (38).
4.2 Signal reconstruction segment analysis

Due to implementation requirements for resampling a row/column, both \([c_r, c_r]_t\) and \([c_r, c_r]_i,h\) need to be known for resampling a texture row, and \([c_r, c_r]_i,c\) and \([c_r, c_r]_s\) need to be known for resampling an intermediate column, where:

- \([c_r, c_r]_t\) : horizontal signal reconstruction segment on a row in texture space
- \([c_r, c_r]_i\) : horizontal signal reconstruction segment on a row in intermediate space
- \([c_r, c_r]_i,c\) : vertical signal reconstruction segment on a column in intermediate space
- \([c_r, c_r]_s\) : vertical signal reconstruction segment on a column in screen space

The derivations of these segments will be analysed next.

4.2.1 Horizontal pass signal reconstruction segment analysis

The horizontal signal reconstruction segment \([c_r, c_r]_t\) and the horizontal signal reconstruction segment \([c_r, c_r]_i,h\) on a row \(v=v\) can be derived from (35), restated here for convenience:

\[
I_{\text{hit}}(x, v) = \int_{\forall x \in A_{r,c}(x)} h_{p,r}(x - x) \cdot B_r(x, v) \cdot I_{r,h}(x, v) \cdot dx
\]

(42)

It can be observed that the signal reconstruction segment \([c_r, c_r]_t\) is equal to the polygon span \([c_r, c_r]\) in texture space on the row \(v=v\), and the signal reconstruction segment \([c_r, c_r]_i,h\) in intermediate space on the row \(v=v\) (see also Figure 9).

4.2.2 Vertical pass signal reconstruction segment analysis

The vertical signal reconstruction segment \([c_r, c_r]_i,c\) and the vertical signal reconstruction segment \([c_r, c_r]_s\) on a column \(x=x\) can be derived from (31), (34) and (33), all restated here for convenience:

\[
I_{\text{hit}}(x, v) = \int_{\forall x \in A_{r,c}(x)} h_{p,r}(x - x) \cdot B_r(x, v, r) \cdot I_{r,c}(x, v) \cdot dx
\]

(43)

\[
I_{r,c}(x, v) = \sum_{\forall y \in A_{r,c}(y)} h_{p,c}(v - v) \cdot I_{\text{hit}}(x, \overline{y})
\]

(44)

\[
I(x, y) = \int_{\forall y \in A_{r,c}(y)} h_{p,c}(y - y) \cdot I_{r,c}(x, y) \cdot dy
\]

(45)

Where \(I_{r,c}(\overline{x}, y) = I_{r,c}(\overline{x}, \overline{y})\)

Note that for this analysis to be more correct, the intermediate sample intensities should be assumed to be computed using (43) rather then (35) which will be used by the implementation. In order to determine the signal reconstruction segment \([c_r, c_r]_i,c\) on the intermediate column \(x=x\), it should be analysed where the vertical reconstructed signal on the intermediate column is non-zero. Analysis of (43) and (44) shows that vertical signal reconstruction on an intermediate column \(x=x\) is only non-zero for the \(v\)-positions for which the corresponding horizontal polygon spans overlap the horizontal prefilter footprint extend centered around \(x=x\):

\[
< \forall v \in \mathbb{R} : \exists \mathcal{A} \in A_{r,c}(\overline{y}) : B_r(x, v) = I > : v >
\]

(46)
As an example, in Figure 10 the signal reconstruction segments \( \{ e_{rs}, e_{vr} \} \), (green lines) in intermediate space are shown for all the intermediate columns. For the intermediate column \( x = \tilde{x} \), the horizontal prefilter footprint extend is explicitly shown.

Figure 10 Vertical signal reconstruction segment on the intermediate columns

(46) can be understood by analysing the vertical reconstruction at a position \((\tilde{x}, v_{rl})\) on the intermediate column: The horizontal signals used in the computation of the intermediate samples which are required for vertical reconstruction at \((\tilde{x}, v_{rl})\) are bounded by the horizontal polygon span on \( v = v_{rl} \). Assume now that for \((\tilde{x}, v_{rl})\) the horizontal polygon span on \( v = v_{rl} \) overlaps the horizontal prefilter footprint extend of the intermediate resample column \( x = \tilde{x} \). In this case, the bounded signals on the intermediate rows will overlap the prefilter footprints of the intermediate samples. Therefore, the intermediate samples will be non-zero which will result in a non-zero vertical reconstructed intensity. If for \( v = v_{rl} \) the horizontal polygon span does not overlap the horizontal prefilter footprint extend of the intermediate resample column \( \tilde{x} \), the reconstructed intensity will be zero due to the intermediate samples now being zero. Using this analysis, it can be derived that the vertical reconstruction segments on the intermediate columns are exactly enclosed by an extended version of the polygon in intermediate space, the extension being equal to half the prefilter width in horizontal direction to both the left and right side (Figure 10).

It can be assumed for now that \( \{ e_{rs}, e_{vr} \} \) is obtained by warping \( \{ e_{rs}, e_{vr} \} \) according to the warp function for the vertical pass.
4.3 Resampling Implementation

K. Meinds has researched an efficient 1D resampling implementation for use in a two pass forward texture mapping implementation [12]. This resampling implementation is used for computing the intensities at the intermediate samples / pixels in a row / column based manner according to the two pass forward texture mapping formulas (15) and (16). This implementation is however not capable of correctly generating edge anti-anti aliased images, basically implementing the behaviour of resampling a warped 2D finite signal reconstructed from a 2D finite discrete signal. Since the resampling implementation required for generating edge anti-aliased images is an extension to the resampling implementation by K. Meinds, the resampling implementation by K. Meinds will be briefly discussed here first. The resampling implementation for generating edge anti-aliased images will be discussed in 4.3.2.

4.3.1 Resampling implementation by K. Meinds

The resampling implementation differs from the computation model as described in paragraph 2.2 in the sense that it warps the discrete samples of the discrete signal on a source row/column onto the destination row/column requiring resampling, reconstructing the signal after being warped. The 1D resampling implementation of K. Meinds is based on the following observations by [11]:

- If the warp magnifies the discrete signal, signal reconstruction should be performed using a quality reconstruction filter, while prefiltering at the resample positions can be ignored.
- If the warp minifies the discrete signal, signal reconstruction can be performed using a trivial reconstruction filter, while a quality prefilter is required at the resample positions.

Both the signal minification and magnification case will be briefly discussed next.

4.3.1.1 Signal minification resampling

Because the signal is minified, trivial signal reconstruction using a box filter is performed. An example of trivial signal reconstruction using a box filter is shown in Figure 11. To arrive at an intensity value for the resample position shown, the part of the reconstructed signal inside its prefilter footprint is weighed with the prefilter profile.

![Figure 11, Resampling in case of signal minification](image-url)
4.3.1.2 Signal magnification resampling

Since the signal will be point sampled at the resample locations only and prefiltering is not necessary in a magnification situation, the quality reconstructed signal needs only to be known at the resample positions. In order to be able to perform quality reconstruction at a resample position, the reconstruction filter profile needs to be warped to the resample domain as well. An example of reconstruction in case of signal magnification can be seen in Figure 12. In Figure 12 a warped reconstruction filter is shown, its width magnified according to the magnification factor. The intensity value at the resample position is obtained by weighing the four warped samples with the reconstruction filter profile.

![Warped Reconstruction Filter](image)

Figure 12, Resampling in case of signal magnification

4.3.2 Edge anti-aliasing resampling implementation

A 1D resampling implementation is now to be derived which computes the intensity contributions and blend fractions according to the two pass forward mapping formulas (35) and (33). The resampling implementation will be used for computing the the intensity contributions and blend fractions at the intermediate samples / pixels in a row / column based manner. In general terms, this requires the resampling implementation to be able to correctly resample a reconstructed signal bounded to a signal reconstruction segment $[r_c, c_r]$, where $[r_c, c_r]$ represents the signal reconstruction segment on the resample row/column.

The following notation will be used for abstracting from the actual pass:
- $[r_c, c_r]_a$ : the signal reconstruction segment on the row/column containing the samples to be warped
- $[r_c, c_r]_b$ : the signal reconstruction segment on the resample row/column

Furthermore:
- $W_p$ : prefilter width
- $W_r$ : quality reconstruction filter width

The following restriction is imposed:
- In order for the resampling implementation to be able to reuse the implementation by K.Meinds without requiring extensive modification, it is desired that the resampling implementation should ideally try to use only either a quality prefilter or a quality reconstruction operation at a resample position.

Both the signal minification and the signal magnification resampling implementations are presented next.

4.3.2.1 Signal minification resampling implementation
In a signal minification situation, the discrete signal (intensity / blend fraction) can be trivially reconstructed similar to the signal minification case presented in paragraph 4.3.1.1. In Figure 13 a minification situation is shown where the discrete signal is trivially reconstructed inside the signal reconstruction segment \([e_n, e_m]_b\). Note that this requires the trivial reconstruction at \((e_n)_h / (e_m)_h\) to be 'cropped'. A prefilter needs only to be applied at the resample positions since resampling only occurs at these positions. Furthermore, resampling is only useful for resample positions for which the prefilter overlaps the signal reconstruction segment \([e_n, e_m]_b\). A resample positions for which the prefilter footprint overlaps the signal reconstruction segment entirely will be referred to as an \textit{interior resample position}, and a resample positions for which the prefilter footprint only partially the signal reconstruction segment an \textit{edge resample position}.

![Figure 13, Resampling in a minification situation.](image)

Because trivial reconstruction is performed using a box reconstruction filter, the first sample \(S_n\) required to be warped for signal reconstruction in the worst case scenario is given by:

\[
S_n = \lfloor (e_n)_h \rfloor
\]  

(47)

Similarly, the last sample \(S_c\) required to be warped for signal reconstruction in the worst case scenario is given by:

\[
S_c = \lfloor (e_m)_h \rfloor
\]  

(48)

This minification resampling scheme can be readily incorporated into the minification resampling implementation by K.Meinds requiring only a minor modification in order to crop the trivially reconstructed signal at \((e_n)_h / (e_m)_h\).

\[4.3.2.2 \quad \text{Signal magnification resampling implementation}\]

In Figure 14, a signal magnification situation is shown. The signal magnification resampling implementation presented in paragraph 4.3.1.2 stated that in signal magnification the prefilter at a resample position can be safely ignored. This will show to be only partially true for this resampling implementation: At an interior resample position, the prefilter can indeed be safely ignored, since the blend fraction is equal to 1. However, for edge resample positions the prefilter is required in order to obtain a properly blended intensity contribution / blend fraction. Since for signal magnification the signal needs to be quality reconstructed, the blended intensity / blend fraction at such a resample position should ideally be obtained by applying a quality prefilter on the continuous quality reconstructed signal. This scheme can not be readily implemented since for these resample positions it would require quality reconstruction to be performed on a \textit{continuous} segment. A practical implementation is therefore to be derived.
Two solutions are now proposed:

(1) The continuously quality reconstructed signal is approximated by the trivial reconstruction on a set of quality reconstructed intensities (Figure 15). This would however require extensive modification to the resampling implementation and it is furthermore an expensive way to arrive at the blended intensity contribution / blend fraction at an edge resample position.

(2) The quality reconstruction in the prefilter footprints is replaced with trivial signal reconstruction (Figure 16). The quality will be less than for method (1), but it is still expected to be visually acceptable. This scheme can be implemented by simply forcing resample positions for which the prefilter footprint only partially overlaps the reconstructed signal to be resampled using the signal minification scheme presented in paragraph 4.3.2.1. This scheme was therefore chosen to be implemented. Further analysis will therefore assume this scheme.
In the worst case scenario, the required samples are determined by the quality reconstruction occurring for the interior pixels. The first sample $S_1$ required to be warped for signal reconstruction is determined by:

$$S_1 = \lfloor (e_{\alpha})_h \rfloor - 0.5*W,$$

(49)

Similarly, the last sample $S_2$ required to be warped for signal reconstruction is determined by

$$S_2 = \lceil (e_{\alpha})_h \rceil + 0.5*W,$$

(50)

Figure 16, Approximating quality reconstruction using trivial reconstruction
4.4 Implementation details

4.4.1 Horizontal pass

For resampling a row in the horizontal pass, the signal reconstruction segment \([e_n, e_o]\) is required for determination of the samples and the signal reconstruction segment \([e_n, e_o]_h\) is required for correct signal bounding during resampling. The signal reconstruction segment \([e_n, e_o]\) in texture space is determined by the horizontal polygon span \([e_n, e_o]\) and the signal reconstruction segment \([e_n, e_o]_h\) is determined by the horizontal polygon span \([e_n, e_o]\) (see 4.2.1). A horizontal polygon span \([e_n, e_o]\) in texture space can be determined using scan conversion [9]. The scan conversion scheme for determination of \([e_n, e_o]\) will actually be imposed by the scheme chosen for vertical signal reconstruction segment determination (see 4.4.2). Vertical signal reconstruction segment determination in intermediate space is more complex since intermediate space is generally curved, the curvature depending on the warp. This also complicates the determination of the horizontal signal reconstruction segments \([e_n, e_o]_h\) in intermediate space since they cannot be readily determined by regular scan conversion. A signal reconstruction segment \([e_n, e_o]_h\) will therefore be obtained by warping \([e_n, e_o]\) according to (2). This can be combined with warping the samples which are also warped according to (2). Whether this is viable will depend on the scheme chosen for vertical signal reconstruction segment determination.

4.4.2 Vertical pass

The vertical signal reconstruction segments \([e_n, e_o]_h\) in intermediate space cannot be readily be determined by regular scan conversion. This is due to the fact that intermediate space is generally curved, the curvature depending on the warp. Three approaches for determination of the vertical signal reconstruction segments \([e_n, e_o]_h\) and \([e_n, e_o]_v\) will be discussed next.

4.4.2.1 Texture space based vertical signal reconstruction segment determination

Consider determining the vertical signal segment \([e_n, e_o]_v\) on the intermediate column \(x = x\) in Figure 17. Only the top boundary \((e_n)_v\) of the vertical signal segment \([e_n, e_o]_v\) will be analysed since determination of the bottom boundary is derived in a similar fashion. The intermediate column \(x = x\) can be seen to intersect the 'virtual' edge \(E_{a}\) from the extended polygon which encloses all the vertical signal reconstruction segments (see 4.2.2). Computing the intersection of the intermediate column \(x = x + 0.5 \times W_p\) with the 'virtual' polygon edge \(E_{a}\) can be seen to result in the same \((x, v)\) coordinate. This intersection problem can be solved by expressing \(v\) in (51) as a function of \(x\), where \(u\) is to be substituted by the texture spaced based linear edge equation \(u = R v + C\) of polygon edge \(E_{a}\):

\[
\begin{align*}
x + 0.5 \times W_p &= F_a(u, v) = \frac{Au + Bv + C}{Gu + Hv + I} \quad (51) \\
v &= \frac{a_1 x + a_2}{a_3 x + a_4} \quad (52)
\end{align*}
\]

The resulting formula expressing \(v\) as a function of \(x\) is of the form:
Figure 17, Determination of the vertical signal reconstruction segment boundary ($E_{re}$).

For a specific $\bar{x}$, $v$ will now represent ($E_{re}$)$_{\bar{x}}$. Obtaining ($E_{re}$)$_{\bar{x}}$ depends on the scan conversion & midpoint scheme which will be used (see paragraph 4.4.2.1.1). Note that it was not mentioned how an implementation would decide which 'virtual' edge to use. This can be determined by first checking whether $\bar{x}$ falls to the left / right with respect to the top intermediate vertex of the polygon. The polygon edge to be used can now be determined by checking which top polygon edge overlaps the column $x = \bar{x} + /- 0.5*W_p$. Note that for intermediate columns near the polygon top vertex no corresponding top polygon edge exists. The top boundary in this case can however taken to be equal to the $v$-coordinate of the polygon top vertex. The implication of this method for determining which scan conversion scheme to use will be discussed next.

4.4.2.1.1 Scan conversion and midpoint considerations

This scheme suggests DDA based scan conversion to be used since the slopes of two current edges are required in order to solve $[\ell_{re}, \ell_{re}]$, as a function of $\bar{x}$ and furthermore edge switches should be kept track of. The incremental computation of the DDA scan conversion will however not be used in the vertical pass. For determination of $[\ell_{re}, \ell_{re}]$, formula (52) can be implemented using two midpoint algorithm implementations (5) [1],[4]. The intermediate sample and $[\ell_{re}, \ell_{re}]$, boundary warping to screen space could be combined requiring only one midpoint algorithm implementation based on (6). Combining this would require the step size of $v$ traversing an intermediate column to be sub-unity advancing to either a sample - edge (52)$_{\bar{x}}$, or edge (52)$_{\bar{x}}$ - sample situation in order to accurately determine $[\ell_{re}, \ell_{re}]$, in case of vertical signal magnification. This sub-unity step size could be dynamically adjusted according to the magnification factor, and could be optimized by incorporating several different step sizes. The selected precision error $\delta$ of the midpoint implementation computing $[\ell_{re}, \ell_{re}]$, in this case is to be selected according to the desired precision for $[\ell_{re}, \ell_{re}]$. For error analysis of $\ell_{re}$ (the analysis for $\ell_{re}$), being similar) assume the midpoint implementation based on (6) computing $y$ to have a selected precision error of $\varphi$, and that this midpoint algorithm uses a sub-unity step size $\eta$ for the sample - edge (52)$_{\bar{x}}$, or edge

---

5 The midpoint algorithm allows to compute the output of a function containing multiplications or divisions for sequence of input values incrementally through iteration rather than using multiplications and divisions for each input value. The precision error in the output value can be selected to a desired precision.
sample situation. The maximum precision error in \( (c, r) \), would then be equal to \( M_1(\delta + \eta) + \varphi \) assuming a maximum vertical magnification factor \( M \) (going from intermediate space to screen space) for which is desired that no noticeable artifacts are displayed. The parameters \( \delta, \eta \) and \( \varphi \) should be chosen such that the maximum precision error in \( (c, r) \), is acceptable.

For the horizontal pass the DDA scan conversion implementation would compute \( [c_r, c_r] \) incrementally. Assume the slopes required for the incremental span computation of \( [c_r, c_r] \) to be implemented using a fixed point register with a fraction of \( N \) bits, and \( K \) to be the maximum number of incremental computations possible (representing the case where no edge switch and therefore no realisation of the slope has occurred), then the precision error in the incremental computation of the \( K \)th \( (c_r, c_r) \) is equal to \( K(0.5)^{N+1} \). The two midpoint implementations which determine \( [c_r, c_r] \) in the vertical pass can be reused for determination of \( [c_r, c_r] \), according to (2). For each midpoint implementation this would require \( n \) to be substituted with the linear edge equation \( n = Rv + C \) of one of the two current polygon edges. Assuming a selected midpoint precision error of \( \beta \), the error in \( (c_r, c_r) \), becomes \( M_1K(0.5)^{N+1} + \beta \) after advancing \( K \) rows. Note that the error part \( M_1K(0.5)^{N+1} \) is due to the midpoint implementation implicitly computing \( (c_r, c_r) \), incrementally by itself (due to the linear edge being substituted) using the slope with the precision of \( N \) bits. The midpoint implementation for warping the intermediate samples and \( [c_r, c_r] \) in the vertical pass can be reused for texel warping in the horizontal pass.

An alternative approach less prone to errors for determination of \( (c_r, c_r) \), (the approach for \( (c_r, c_r) \), being similar) in the vertical pass would be to implement a midpoint algorithm separately for direct determination of \( (c_r, c_r) \). This requires a midpoint implementation of the equation which expresses \( (c_r, c_r) \), directly as a function of \( \tau \). For example, the equation which expresses \( (c_r, c_r) \), directly as a function of \( \tau \) for the columns intersecting edge \( E_a \) can be obtained by substitution of (52) into the vertical warp function (6). The error in \( (c_r, c_r) \), is now no longer dependant on \( \delta \) and \( \eta \). The disadvantage is that this alternative approach requires two additional midpoint implementations. A small advantage would however be gained again, since \( [c_r, c_r] \), would no longer be needed to be warped together with the midpoint implementation warping the intermediate samples. If the alternative option is used, the midpoint implementation warping intermediate samples would be reused for texel warping in the horizontal pass. The two midpoint implementations determining \( [c_r, c_r] \), in the vertical pass would be unused in the horizontal pass.
4.4.2.2 Screen space based vertical signal reconstruction segment determination

The screen space based determination approach determines the vertical screen space signal reconstruction segments \([e_o, e_r]\), of the extended polygon enclosing all the vertical signal reconstruction segments in screen space using regular scan conversion. This is possible since the extended polygon edges in screen space are straight lines (16). A vertical signal reconstruction segment \([e_o, e_r]\) is warped back to intermediate space generating \([\tilde{e}_o, \tilde{e}_r]\) according to \(H(x,y)\):

\[
H(x,y) = \frac{Nx + Py + Q}{Rx + Sy + T} \tag{53}
\]

The required intermediate samples can then be determined by using \([\tilde{e}_o, \tilde{e}_r]\). These intermediate samples are then warped to screen space according to (6). In order to obtain the extended polygon in screen space, the polygon vertices must first be extended in intermediate space by half the prefilter width in the horizontal direction, generating the vertices of the extended polygon enclosing the signal reconstruction segments in intermediate space. The vertices of the extended polygon in intermediate space are then warped to screen space after which scan conversion can be used. Note that it is unfortunately not possible to implement a scheme similar to texture space based approach using the polygon edges for derivation of \([e_o, e_r]\). This is because the extended polygon edges in screen space are perspective shifted versions of the polygon edges.

![Figure 18 The extended polygon in screen space enclosing the vertical signal reconstruction segments](image-url)

Figure 18 The extended polygon in screen space enclosing the vertical signal reconstruction segments
4.4.2.2.1 Scan conversion & midpoint considerations

The segment \( \{e_{on}, e_{of}\} \) can be determined by a DDA scan conversion implementation. An edge seeking traversal algorithm for determination of \( \{e_{on}, e_{of}\} \), according to Pineda [15] is inefficient since it will traverse the entire extended polygon interior in stead of staying close to the edge. An edge seeking traversal algorithm is however feasible when two traversers are used, one for each boundary. Both scan conversion schemes will be discussed next, for which midpoint integration will be analysed.

4.4.2.2.1.1 DDA based scan conversion

Given the two current polygon edges on the current pixel column, the DDA scan conversion computes the signal reconstruction segments \( \{e_{on}, e_{of}\} \), incrementally, the precision error in \( e_{on} \) / \( e_{of} \), for the \( K \)'th column being equal to \( K(0.5)^{K^{+1}} \). Two midpoint implementations determine \( \{e_{on}, e_{of}\} \), by implementing \( v = H(x, y) \), substituting \( y \) with one of the screen space based linear polygon equations of the form \( y = R + C \). Both midpoint implementations therefore implicitly compute \( \{e_{on}, e_{of}\} \) internally for each \( x \) while determining \( \{e_{on}, e_{of}\} \). Note that the substitution requires the slopes which are computed by the DDA implementation. The substitution allows the midpoint implementations to be solely dependant on \( x \), which eliminates the need for the midpoint implementations to vertically iterate to \( \{e_{on}, e_{of}\} \). For error analysis of \( \{e_{on}, e_{of}\} \) (the analysis for \( \{e_{on}, e_{of}\} \), being similar), assume the selected midpoint precision error in computing \( \{e_{on}, e_{of}\} \), to be \( \delta \). The error in \( \{e_{on}, e_{of}\} \), implicitly used in the midpoint implementations after having advanced \( K \) columns is equal to \( K(0.5)^{K^{+1}} \). This error is magnified with \( M^{K} \) and combined with error \( \delta \) due to the selected midpoint precision error. The total error in \( \{e_{on}, e_{of}\} \), is therefore equal to \( M^{K} K(0.5)^{K^{+1}}+\delta \). In a magnification situation, \( M^{K} \) is large and therefore the error \( M^{K} K(0.5)^{K^{+1}}+\delta \) is (relatively speaking) large. This can cause the required intermediate sample determination using \( \{e_{on}, e_{of}\} \), to be off by several samples. This does however not matter since the contribution of these samples to an edge pixel will be small in magnification. The error in the signal reconstruction segment boundary \( \{e_{on}, e_{of}\} \), in the prefilter footprint of an edge pixel being fixed to \( K(0.5)^{K^{+1}} \). In a magnification case, sample determination is required to be accurate. This is again not a problem since \( M^{K} \) is now small and \( M^{K} K(0.5)^{K^{+1}}+\delta \) as well. Besides the two midpoint implementations for determining \( \{e_{on}, e_{of}\} \), a separate midpoint implementation is required for warping the required intermediate samples to screen space.

This DDA & midpoint scheme can be reused in the horizontal pass: The DDA scan conversion determines \( \{e_{on}, e_{of}\} \) incrementally and the two midpoint implementations as described are used for determination of \( \{e_{on}, e_{of}\} \), similar to 4.4.2.1.1). The midpoint implementation warping intermediate samples is reused for warping the texels to intermediate space. The number of fractional bits \( N \) for the slope should be based on keeping the error \( M^{K} K(0.5)^{K^{+1}}+\delta \) in \( \{e_{on}, e_{of}\} \), to within a desired precision in the horizontal pass.

4.4.2.2.1.2 Edge seeking traversal based scan conversion

In this case, the two midpoint algorithms must implement \( H(x, y) \) in order to determine \( \{e_{on}, e_{of}\} \), since the edge slopes are not known. Each midpoint implementation should therefore iterating along with the one of the two traversers to \( \{e_{on}, e_{of}\} \). For error analysis, assume the selected midpoint precision error in computing \( \{e_{on}, e_{of}\} \), to be \( \delta \), and the step size of the traversal to be \( \eta \) in locating an edge. The precision error in \( \{e_{on}, e_{of}\} \), will then be equal to \( \eta \) and the precision error in \( \{e_{on}, e_{of}\} \), will be equal to \( \eta M^{K}+\delta \). Note that the magnification factor \( M^{K} \) doesn't present a problem, similar as described in paragraph 4.4.2.2.1.1). Besides the two midpoint implementations for determining \( \{e_{on}, e_{of}\} \), a separate midpoint implementation is required for warping the required intermediate samples to screen space.

The edge seeking traversal & midpoint scheme can be reused in the horizontal pass: The edge seeking traversal determines \( \{e_{on}, e_{of}\} \) while the midpoint implementations iterate along, determining \( \{e_{on}, e_{of}\} \). The

\[ \delta \]

An edge traversal implementation could incorporate multiple step sizes. It is assumed here that if it detects having crossed an edge it will back up and locate the edge using a final smallest step size of \( \eta \).
midpoint implementation warping intermediate samples in the vertical pass is reused for warping the texels to intermediate space. The precision error in \([\varepsilon_n, \varepsilon_o]_h\) is equal to \(\eta\) and the precision error in \([\varepsilon_{n+}, \varepsilon_{o-}]_h\) is equal to \(\eta M + \delta\). In the horizontal pass \(\eta\) should be based on keeping the error \(\eta M + \delta\) in \((\varepsilon_{n+})_h / (\varepsilon_{o-})_h\) to within a desired precision.
4.4.2.3 Resampling based vertical signal reconstruction segment determination

This approach has not been fully researched. Therefore, only the concept and further work will be discussed.

The idea of resampling based determination is based on the fact that it should be possible to obtain the vertical signal reconstruction segment $[e_u, e_d]$, by resampling of the polygon edges. In order to do so vertical scan conversion is to be performed in texture space, which would result in a vertical polygon span for each texture column. An example of a vertical polygon span $[e_u, e_d]$, is given for column $u = \bar{u}$. Now consider the bottom boundaries of the polygon spans for edge $E_a$ represented by the red dots. These can be warped according to:

$$[x, v] = [F_x(u, v), v]$$

The result of warping the boundaries to intermediate space can be seen in Figure 19. Note that the boundaries will generally not coincide with the intermediate sample positions.

![Figure 19: Warping the boundaries](image)

The warped boundaries can actually be thought of being a discrete signal, for which the $v$-coordinates now represent the 'intensities' of 'samples'. Consider this discrete signal to be reconstructed (without being prefiltered) and sampled at the intermediate resample column $x = \bar{x}$ (Figure 20). This results in the bottom boundary $(e_d)_i$, of the polygon span on the intermediate sample column $x = \bar{x}$. This bottom boundary on $x = \bar{x}$ represents the bottom boundary $(e_d)_i$, of the vertical reconstruction segment on $x = \bar{x} - 0.5*W_p$. After processing each polygon edge in texture space in a similar fashion, all the vertical reconstruction spans on intermediate columns will be known except on the intermediate columns near the polygon top/bottom vertex. These can however be taken equal to the $v$-coordinate of the top/bottom vertex. The vertical signal reconstruction segments in screen space can be found by warping of the vertical reconstruction spans in intermediate space.
Some issues requiring further research would be:

- A bottom polygon edge in texture space is not guaranteed to be a bottom polygon edge in intermediate space (see edge $E_b$ in Figure 21). This would make resampling of bottom polygon edges in one pass more complicated. Even if this would be resolved an other problem might be introduced, since it is expected that the resampling of $v$-coordinates near edge transitions will be inaccurate.

- Assuming resampling on a per edge basis, reconstruction near the edge of a polygon edge requires additional artificially introduced samples. This would for example require the polygon edges to be extended in texture space before being vertically scan converted.

Figure 20 Reconstruction and sampling of the 'boundary signal'

Figure 21 polygon edge $E_b$ switches from being a bottom edge to a top edge
4.4.2.4 Vertical signal reconstruction segment determination summary

The screen space based determination method allows for a simple implementation using 3 midpoint implementations. This method is favoured over the texture spaced based method since the combined sample and edge warping approach would require high precision in the vertical pass in case of vertical magnification and the alternative using 5 midpoint implementations offers no advantage over the screen space based determination method.

The screen space based determination method can either be implemented using DDA based scan conversion or edge seeking traversal based scan conversion. Whether the screen space based approach is to be implemented in hardware using a DDA scan conversion based implementation or an edge seeking traversal scan conversion based implementation would require further research which is considered beyond the scope of this thesis. For testing purposes, the screen space based determination approach using DDA scan conversion was implemented without using the midpoint algorithm for implementing the warp functions. The test results are presented next.
4.5 Test results

The next pages show the result of texture mapping a triangle for respectively a 2x minification case, 2x magnification case and 6x magnification case. All images were generated using a cubic filter (4 samples) for prefiltering / reconstruction. Note that the magnification cases show serious artefacts near the edges, the artefacts worsening as the magnification factor increases. The cause of these artefacts will be analysed more in depth in paragraph 4.5.
The quick brown fox jumps over a lazy dog.
The quick brown fox jumps over a lazy dog.
4.6 Error Analysis

Figure 20 shows a closeup near an edge in the final image for a magnification case. Three artefacts can be seen:
1. the intensity gradually fades in moving away from the edge
2. the intensity is too low
3. the intensity is too bright

Figure 22 Closeup near an edge in the final magnified image

Artefact (1) occurs at the interior pixels and is due to the approximation of (55) with the implementation formula (56):

Intensity of an intermediate sample $\bar{x}, \bar{v}$ (theoretical):

$$I_{\text{int}}(\bar{x}, \bar{v})_{\bar{x}, \bar{v}} = \int_{\forall \gamma \in \mathcal{A}_{\gamma, \bar{v}}(\bar{x})} h_{\gamma, y}(\bar{x} - x) \cdot B_\gamma(x, \gamma) \cdot I_{\gamma, A}(x, \gamma) \cdot dx$$  \hspace{1cm} (55)

Intensity of an intermediate sample $\bar{x}, \bar{v}$ (implementation):

$$I_{\text{int}}(\bar{x}, \bar{v})_{\gamma \in \mathcal{A}_{\gamma, \bar{v}}(\bar{x})} = \int_{\forall \gamma \in \mathcal{A}_{\gamma, \bar{v}}(\bar{x})} h_{\gamma, y}(\bar{x} - x) \cdot B_\gamma(x, \gamma) \cdot I_{\gamma, A}(x, \gamma) \cdot dx$$  \hspace{1cm} (56)

Due to the approximation formula (56) being used, the relative intensity error of an intermediate sample $\bar{x}, \bar{v}$ used for vertical reconstruction in the vertical pass at a position $\bar{x}, \bar{v}$ is determined by (see also par 4.1):

$$E_{\gamma, A}(\bar{x}, \bar{v}) = \int_{\forall \gamma \in \mathcal{A}_{\gamma, \bar{v}}(\bar{x})} h_{\gamma, y}(\bar{x} - x) \cdot \{B_\gamma(x, \gamma) - B_\gamma(x, \gamma)\} \cdot dx$$  \hspace{1cm} (57)
In a signal magnification case, only quality reconstruction is used at an interior pixel (see 4.3.2.2). The relative error in the pixel intensity is therefore determined by the relative error in the vertically quality reconstructed intensity. The relative error in an vertically reconstructed intensity at a certain \((\bar{x},v)\) (which will be assumed to map onto the pixel of interest) is determined by:

\[
E_{rr}(\bar{x},v) = \sum_{v \in A_{r,v}} E_{r,v}(\bar{x},v) \cdot E_{rr}(\bar{x},v)
\]

(58)

Analysing (57), it can be seen that the relative error of an intermediate sample intensity used for reconstruction is determined by the mismatch in the horizontal signal reconstruction segment area its prefilter footprint. The mismatch is due to using the horizontal polygon span on its row for determination of the horizontal signal reconstruction segment (56), rather than using the horizontal polygon span on the vertical reconstruction position (55). Consider now Figure 23 which represent the resampling situation for column \(x = \bar{x}\) as seen in Figure 20. For almost all the interior pixels shown in Figure 23 (note that interior is defined with respect to the vertical signal reconstruction segment, see 4.3.2.2), the horizontal polygon spans on their corresponding rows will totally overlap the prefilter footprint extend centered around \(x = \bar{x}\).

The horizontal signal reconstruction segments on the rows of the four intermediate intensities in case of correct reconstruction at these pixels will therefore also completely overlap their prefilter footprints.

- intermediate sample
- edge pixel
- interior pixel

---

Figure 23: Vertical resampling on pixel column \(\bar{x}\)

The horizontal signal reconstruction segments on the rows of the four intermediate intensities in case of the implemented reconstruction scheme are however taken to be equal to the horizontal polygon spans on their
rows. For the intermediate samples \((\bar{X}, \bar{y})\) and \((\bar{X}, \bar{y}2)\), the horizontal polygon spans on their rows do not overlap their prefilter footprints, the relative intensity error of these samples being used for quality reconstruction at the interior pixels therefore being equal to \(-1\) according to (57). For the intermediate samples \((\bar{X}, \bar{y})\) and \((\bar{X}, \bar{y}0)\), the horizontal polygon span on their rows completely overlap their prefilter footprints, the relative intensity error of these samples being used for reconstruction at the interior pixels therefore being equal to \(0\) according to (57). The relative error in the quality reconstructed intensity at a pixel is now determined by weighing the relative errors with to the reconstruction filter according to (58). As an example, the reconstruction filter is shown in Figure 23 for the pixel \((\bar{x}, \bar{y})\). For interior pixels further away from the edge, the relative errors of \((\bar{X}, \bar{y})\) and \((\bar{X}, \bar{y}2)\) will be weighed less. The side plot in Figure 23 shows the theoretical vertically reconstructed signal versus the actual reconstructed signal for all positions on the pixel column. This plot is obtained by assuming \(L_{\text{tr}}(\bar{x}, \bar{y})\) to be one in (55) and (56), expressing the final reconstructed intensity using (34) in percentages rather than a fraction. The reconstructed intensity as a result of using the implementation formula (56) for computing the intermediate sample intensities can be seen to gradually fade in moving away from the polygon edge.

Figure 24 (closeup of Figure 23) shows the situation for the edge pixels containing artefact 2. An edge pixel in case of signal magnification is resampled using trivial signal reconstruction and a prefilter positioned at the pixel (see 4.3.2.2). Consider the edge pixel \((\bar{x}, \bar{y})\). It can be seen that the final pixel intensity will be zero, since the trivially reconstructed signal is zero inside the prefilter footprint (Figure 24, sideplot b). The theoretical reconstructed signal which should have been used is shown in Figure 24, sideplot a.

![Figure 24 Edge pixel resampling situation](image)

Figure 24 Edge pixel resampling situation
In Figure 25 the situation is shown for the column $x = \bar{x}$ in Figure 20. Consider the edge pixel $(\bar{x}, \bar{y})$. It can be seen that the trivially reconstructed signal now represents an overestimation of the theoretical reconstructed signal. The final pixel intensity will be too bright, resulting in artefact 3.

Figure 25 Edge pixel resampling situation
4.7 Intensity correction scheme

A correction scheme can be included which corrects the intensities in case of magnification. Since the blend fraction is resampled similar to the intensities, it will suffer from the same computational error. An artificial full intensity can be regained by multiplying the intensities with the inverse blend fraction. This requires an alternative blend fraction computation scheme in order to compute the correct blend fraction which can then be used to blend the intensities correctly. The error formulas can be implemented in order to determine whether a pixel requires intensity correction if its error lies above a certain threshold. However, since the correction scheme presented here was finally only used for testing, no such error threshold scheme has been devised. For testing, all pixels were simply assumed to require intensity correction. An adjust which is required for the correction scheme is discussed next, after which the alternative blend fraction computation method used will be discussed. The alternative blend fraction computation method was also only used for testing and might therefore not be suitable for an actual implementation.

4.7.1 Adjusting trivial reconstruction near signal reconstruction segment boundaries

In order to be able to multiply the intensity with the inverse blend fraction, an adjustment is required to the resampling implementation. This is because some edge pixels will have a zero intensity/blend fraction (artefact 3) which does not allow for inverse multiplication. In order to obtain an intensity/blend fraction at such an edge pixel, the trivial reconstruction of the two intermediate samples lying at each side of a signal reconstruction segment boundary \((e_n)\), \((e_{n+1})\) is adjusted. In Figure 26, two intermediate samples and the signal reconstruction segment boundary \((e_n)\), are shown on a pixel column. The intermediate sample \((\bar{x}, \bar{v})\) represents the sample lying on the inside, and \((\bar{x}, \bar{v}+1)\) the intermediate sample on the outside. Normally the vertical trivial reconstruction span of the intermediate sample \((\bar{x}, \bar{v})\) is given by \([F(\bar{v}-0.5), F(\bar{v}+0.5)]\), and the vertical trivial reconstruction span of the intermediate sample \((\bar{x}, \bar{v}+1)\) is given by \([F(\bar{v}+0.5), F(\bar{v}+1.5)]\). The reconstruction is now chosen such that the vertical trivial reconstruction span of the intermediate sample \((\bar{x}, \bar{v})\) is given by \([F(\bar{v}-0.5), F(\bar{v}+0.5)]\) and the vertical trivial reconstruction span of the intermediate sample \((\bar{x}, \bar{v}+1)\) is given by \([F(\bar{v}+0.5), F(\bar{v}+1.5)]\). The edge pixel shown in Figure 27 will now receive an intensity and blend fraction. In a similar way, the trivial reconstruction is adjusted near the signal reconstruction segment boundary \((e_n)\).

![Figure 27 Adjusting trivial reconstruction near a signal reconstruction segment boundary](image-url)
4.7.2 Alternative blend fraction computation method

Consider Figure 28 which will be used for analysis. The blend fraction is to be computed for the pixel \((\bar{x}, \bar{y})\) shown.

\[ A(x, y) = \int_{\gamma(y') \in A_{\gamma, x}^{\gamma(x)}} h_{p, \gamma}^{\gamma(x)}(x - x) \cdot h_{p, \gamma}^{\gamma(x)}(\bar{y} - \gamma) \cdot B_\gamma(x, y) \cdot dy \cdot dx \]  

The alternative blend fraction scheme is based upon the separable formula determining the 2D blend fraction in the continuous screen space domain:

\[ \lambda(x, y) = \int_{\gamma(x) \in A_{\gamma, x}^{\gamma(x)}} \int_{\gamma(y) \in A_{\gamma, y}^{\gamma(y)}} h_{p, \gamma}^{\gamma(x)}(x - x) \cdot h_{p, \gamma}^{\gamma(y)}(\gamma - \gamma) \cdot B_\gamma(x, y) \cdot dy \cdot dx \]  

The alternative blend fraction scheme computes the blend fraction in screen space according to using a discrete approximation of the vertical prefiltering: For each vertical polygon span (red), a 1D vertical prefilter is applied (Figure 29a). A prefilter filters a signal of value 1 inside a vertical polygon span and a signal of value 0 outside. The vertical polygons span can be determined on a sub pixel column grid in order to increase precision. The vertical 1D prefilter operations results in a series of 1D vertical blend fractions on the pixel row. The trivial reconstruction of this discrete signal is then horizontally prefiltered resulting in the final blend fraction for the pixel (Figure 29b)

Figure 29 (a) prefilter footprint top view (b) prefilter footprint sideview on the pixel sample row.
4.7.2.1 Test results

The result of the correction scheme is shown in Figure 30, showing both the intermediate- and final image for the 6x magnification case. Although the blending appears to be far better than without correction (final image closeup bottom), the edge still contains artefacts (final image closeup left). This is due to the trivial reconstruction being used in the horizontal pass for computing edge pixels (intermediate image closeup). In order for the correction scheme to work, the intermediate image should contain no artefacts since the correction scheme only corrects the final image. This error can therefore only be corrected by using a better resampling scheme for edge pixels in the magnification case. This could be achieved for example by using scheme (2) presented in paragraph 4.3.2.2.

Figure 30 Intermediate and final image using the correction scheme for a 6x magnification situation
5 Conclusions

Without requiring extensive modification to the existing two pass forward texture mapping implementation by K.meinds it is possible to generate correctly anti-aliased images in texture minification situations, correctly generating a blend fraction for each pixel which is to be used by a composition unit. It is more difficult to generate edge-anti aliased images in case of texture magnification. The main artefact in case of signal magnification is caused by the approximation used for computing the intermediate intensities. This error can however be reduced by using an intensity correction scheme. Such a scheme requires an inverse multiplication with the two pass generated blend fraction and reblending using an alternatively computed blend fraction for a pixel requiring intensity correction. This does not however correct the error made due to the simplified resampling scheme for edge resample positions in a signal magnification situation. In order to reduced these artefacts, higher quality resampling for edge resampling in case of signal magnification is to be used.

5.1 Further work

Some issues requiring further research would be:
- An efficient scheme should be researched for determining when intensity correction is needed
- A more high quality resampling scheme should be researched for edge pixels in a signal magnification case.
- Initial work on a two pass forward texture mapping implementation which computes unblended intensities could be further researched. This implementation bounds the prefiltered signal instead of the reconstructed signal.
References

[1] Barenbrug, B.
Midpoint-based division-free perspective-correct texture mapping
Nat.Lab. Technical Note TN2000/012

Inferring Transforms

[3] Carpenter, L.
The a-buffer, an antialiased hidden surface method

3D Transformations of images in scanline order

[5] Chryssafis, A.
Anti-aliasing of computer-generated images: a picture independent approach.

A comparison of antialiasing techniques

The aliasing problem in computer-generated shaded images.
Communications-of-the-ACM (USA), vo1.20, no.11, p.799-805, Nov. 1977.

[8] Fant, K.M.
A nonaliasing, real-time spatial transform technique
IEEE Computer Graphics and Applications, Jan-86

Computer Graphics : Principles and Practice

High-quality filtering using forward texture mapping.
Computers & Graphics (Pergamon) vol. 15, no. 4, p 569-577,

Texture mapping and resampling for computer graphics
Master’s Thesis, University of Toronto, jan-1991
[12] Meinds, K.
Perspective transformations of images with a FIR-filter
Nat.Lab technical note TN 2000/141

Jagged edges: when is filtering needed?

[14] Peters, F.J. and C.W.A.M van Overveld
Midpoint algorithms for fast perspective correct rendering
Nat.Lab technical note TN 112/99

[15] Pineda, J.
A Parallel Algorithm for Polygon Rasterization

[16] Schilling, A. and W. Strasser
EXACT: algorithm and hardware architecture for an improved A-buffer.
Proceeding of SIGGRAPH 20th Annual International Conference on Computer Graphics
SIGGRAPH '87 Conf Proc, Anaheim, CA, USA, Jul 27-31

[17] Smith, A.R.
Planar 2-pass texture mapping and warping
Computer Graphics (ACM) vol. 21, no. 4, p 263-272, Jul 1987