Chapter 3

Development

> '*All that we can ever comprehend of the Universe must reside within our minds.* [...] *The amazing thing is not that mental models of nature are flawed or diminutive, but that they work at all.*'
>
> (Timothy Ferris, *The Universe and Eye*)

This chapter will present solutions to the various problems that were identified as a result of the preceding literature review. Specifically, we will develop a means to describe any object in a VE in terms of its component spatial frequencies. We will also develop a mathematical model to predict the contrast sensitivity of a standard observer to a stimulus of any velocity and eccentricity. Finally we will produce a framework for polygon simplification which is perceptually predictable and which takes into consideration the surface properties of a model (in addition to the base geometry).

3.1 A Measure for Perceived Detail

3.1.1 Where to Calculate Detail?

One of the first and most critical design decisions which we encounter when building a perceptually-based LOD system is *where* to calculate the perceptual content of an object. We have two choices: either we base our analysis on the (3D) geometric definition of an object (i.e. its polygons and vertices), or the

(2D) rendered image of that object. Both approaches have their distinct pros and cons which we can expand as follows:

Geometrical Description : here we have 3D data available to us, so we can compute the projected 2D spatial frequency profile for any orientation of the model. Also, this could feasibly be computed on-line; during the actual simulation.

However, any such algorithm would probably have to be tied quite closely to the particular rendering model chosen (e.g. polygonal, ray tracing, radiosity, etc.) because it must manipulate the internal representation of an object. Additionally, some visual effects are not described by simply looking at the geometry of a model, e.g. a texture mapped polygon may contain various intricate visual details which would be completely ignored if we only referred to the underlying geometry.

Rendered Image : here we have a 2D snapshot of an object from a single viewpoint. We would therefore have to take multiple snapshots around the object and interpolate between these in order to describe its 3D profile. We would also have to apply the computation off-line because we are required to know the spatial frequency content of a model *before* we display it: once a model has been rendered, we have already expended the computational resources which we were seeking to preserve.

The advantages however are that because we only apply our computation to the final rendered image, the method is independent of the particular rendering technique employed. Also, the rendered image more accurately and completely reflects the visual stimuli which the user is actually presented with.

Based upon the above, it is clear to the author that we must advocate a system which extracts detail from the rendered image of an object rather than its geometrical description. This is because the rendered image is the actual information which is presented to the user's visual system. It is therefore the most accurate indication of perceived detail. Looking at the geometrical description of an object does not give a reliable indication of what the user eventually sees because the geometry can be displayed differently depending upon a number of factors such as: the particular shading model being used, the effect of any light sources, the use of texture maps or environment (reflection) maps, the simulation time of day (TOD), the use of fog or haze effects, transparency, etc.

More ideologically, if we are to develop a system based upon our perception of a scene, we should not have to concern ourselves with *how* the scene was produced, but only with *what* it contains. To use Marr's (1982) terminology, we should be concerned with the *representation* of an image, not the *processes* underlying it.

3.1.2 Spatial Frequency Analysis of Images

We have resolved that we wish to calculate the spatial frequency content of a computer-generated image. Now we have to consider how this analysis may be performed. Essentially, we wish to find a suitable mapping to transform a 2D function of intensity values into a 2D function of spatial frequencies. The solution which we will present here can be broken down into three independent stages. These are listed briefly below and will be further expanded presently.

1. **Feature Extraction** : find all of the atomic visual 'features' in an image. This is perhaps best achieved using an **image segmentation** algorithm with a suitable perceptually-based feature extraction mechanism.

2. **Spatial Frequency Calculation** : calculate all of the relevant spatial frequencies in each feature. At this stage, these frequencies are *relative* because they are in terms of pixels only; with no immediate concept of angular size, i.e. the units of spatial frequency at this stage are cycles per pixel (c/pixel).

3. **Spatial Frequency Transformation** : scale the relative spatial frequency values into units of c/deg. This transformation can be performed once we know the **field of view** (FOV) of the display device.

3.1.3 Review of Feature Extraction Techniques

Before we proceed any further, it would be useful at this point to briefly review some previous attempts which have been made to extract visual features from an image, or to otherwise assess its perceptual content.

There has been a large body of work in the area of **image quality metrics** (see Jacobson, 1995, for a recent review). However this field is dedicated to the concise perceptual evaluation of display devices. As such, these metrics are not concerned with the quality of arbitrary images displayed on these devices; and hence are of little benefit to us here.

Similarly, some work has been done in order to provide a numerical measure of the subjective distance between two images (Rushmeier *et al.*, 1995; Daly, 1993; Mannos and Sakrison, 1974; Gervais *et al.*, 1984), i.e. how perceptually different two images appear. These techniques try to map two arbitrary images to a single scalar quantity which describes how closely the images are perceived as equal. Such methods are therefore not of direct relevance to our problem because they are relative—not absolute—measures, and they do not describe the full range of spatial detail present in an image.

Within the field of computer vision a number of image segmentation algorithms have been founded on perceptual models (see Reed and du Buf, 1993, for a review). Many of these have resorted to Gestalt rules of grouping (Katz, 1951) to resolve all of the perceptually disparate components (e.g. Khan and Giles, 1992). Other feature extraction techniques have included attempts to generate a number of **feature maps** to locate 'meaningful wholes' in an image based upon a number perceptual criteria (e.g. Soufi and Scrivener, 1992), or the use of localised frequency domain techniques to categorise object groupings (e.g. Reed and Wechsler, 1990). In general, all of these algorithms are concerned with the identification of discrete *objects* within a scene, and not with the quantification of absolute detail within the image as a whole. (For example, the extraction of all pixels which describe a hammer; not the finer textural details which can be perceived within that object.) Also, many of these approaches have only been applied to binary or greyscale images: comparatively little consideration has been given to the analysis of full-colour images.

(As an aside, it is also common in computer vision systems to have to deal with noisy, real-world images. A number of techniques have therefore been developed to compensate for, or smooth over, noisy components within an image (Gonzalez and Woods, 1992). However our application does not need to concern itself with noise removal because the source images are computer-generated renderings, not digitised samples, and are therefore noise-free.)

We have already seen that the human visual system is thought to be composed of a number of parallel channels, each sensitive to a particular size of spatial detail. These channels can be modeled using 2D **Gabor functions** (Stork and Wilson, 1990) which are simply the product of a 2D sinusoid and a 2D Gaussian. Some researchers have therefore tried convolving an image using these Gabor functions in an attempt to develop and validate computational models of human vision (e.g. Watson, 1987; Beck *et al.*, 1987). For any single analysis, this convolution will report the degree to which a particular level of detail exists in an image. It does not report the entire range of perceptually visible detail which the image contains. That is, these techniques focus on a different, more basic, problem to ours: i.e. to what degree one particular visual channel is stimulated by a certain amount of detail, whereas we are effectively concerned with *which* channel(s) are optimally stimulated by a certain amount of detail.

No discussion of computational vision would be complete without mentioning the work of the late David Marr. Marr (1982) produced a machine-computable model for early human vision using edge detection techniques which could function over a range of scales. However, although the concepts of feature detection are relevant, this technique is of limited use to us here for the same reason that was given for the Gabor convolutions. Effectively, the size of a stimulus is an input to these detectors, whereas we want a detector which returns the size of a perceived stimulus as its output.

In conclusion, it can be observed that the requirements of our application are subtly different to all of those mentioned above. As such, a novel approach must be formulated. Let us therefore address each of the three stages of spatial frequency calculation which we have just identified in the previous section.

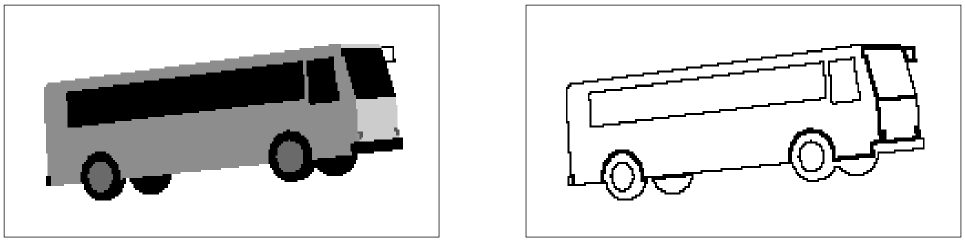
3.1.4 Extracting the Visual Features from an Image

The aim of this first stage is to extract all of the visually atomic 2D features within an image—the absolute elements of detail in the image. From a physiological standpoint, this would be the extent of a region which maximally stimulates a single neural channel in the vision system. Unfortunately, the mechanism which the human visual system uses to decide this delimitation is still unclear. We must therefore formulate our own, albeit simple, model for this process.

3.1.4.1 How to Define the Extent of a Feature

The mechanism which will be adopted here for the task of locating each visual feature in an image is based upon an image segmentation paradigm, i.e. a process which takes an image and segments it into a number of individual regions for independent analysis. This is done by taking a single pixel and then attempting to grow this pixel into a region; merging adjacent pixels with the region based upon a certain **segmentation criterion**. The crux of our dilemma therefore rests in the specification of this segmentation criterion.

The trivial case for defining the extent of a feature would be to only merge pixels which are exactly the same colour. This would be a valid definition if we were using a simple flat-shading polygon renderer (such as the Superscape VRT or REND386 graphics packages); or if we decided to use the flat-shading mode of our graphics renderer for time-efficiency reasons. Figure 3.1 illustrates this simple case of feature extraction.



(a) **(b)**

Figure 3.1: *(a) a flat-shaded image of a bus, and (b) a representation of the boundary of each visual feature in this image. In this case, a visual feature is simply a group of pixels with exactly the same colour.*

However, many contemporary graphics renderers offer far more sophisticated display algorithms than just flat-shading. For example, polygons can often be smooth-shaded, antialiased, and/or texture mapped. We should therefore be able to cope with these more realistic representations in addition to the simple case of flat-shaded primitives. Consequently, it is evident that we must relax the segmentation criterion slightly in order to include colours which are similar to the target colour, but not necessarily exactly the same. But to what extent can we relax this threshold?

In order to define this, we will utilise the body of research into **perceptually uniform colour spaces**, and **just noticeable differences** (JNDs). A perceptually uniform colour space is one in which the numerical distance between two colours (calculated with a suitable **colour difference formula**) is directly related to their perceptual distance. This distance is expressed as a number of JNDs. That is, we have a measure for the degree to which an observer perceives two colours as being distinct.

The reason for adopting this approach is to enable us to locate perceived edges in a colour image. The premise is that in a smoothly shaded region, one does not inherently notice the colour difference between adjacent pixels; but one perceives a gradual colour gradient over the region as a whole. Using JNDs, we can therefore decide whether two adjacent colours form part of a colour gradient (i.e. part of a feature) or are perceived as an edge between features.

3.1.4.2 In Search of a Colour Difference Formula

There is a bewildering array of colour difference formulae to chose from, each differing slightly from the next with regard to some attribute or parameterisation. However, before we converge upon any particular formula, it would be useful to take a cursory look at the background of these systems.

In 1931, the CIE (Commission Internationale de l'Éclairage) proposed the XYZ tristimulus coding scheme to describe colours in a device-independent manner. This provided an accurate and convenient means of specifying an absolute colour, but it did not consider the perceptual relationship between points in the colour space. In fact, the perceptual non-uniformity of the CIEXYZ colour space is about 80:1 (Poynton, 1993), i.e. the discrepancy between the perceptual distance and the numerical distance of two CIEXYZ colours can vary at points by up to about 8,000%.

It was not until 45 years later, in 1976, that the CIE introduced the CIELAB and CIELUV specifications which attempted to provide a more perceptually uniform colour space. These two systems managed to improve the perceptual non-uniformity of the CIEXYZ system to around 6:1. Since that time, a number of other systems have been proposed which try to improve upon the CIE's standard, e.g. the CMC formula (CMC, 1989), the BFD formula (Luo and Rigg, 1987), the FMC1 formula (MacAdam, 1985), etc.

The most relevant colour difference formula for our purposes is the CIELUV system. Most formulae were developed for use in the colourant industry as a product tolerance measure. These systems are therefore designed for viewing under reflected light conditions and are not directly applicable to self-luminous colours such as those generated by visual display units (VDUs). However, the CIELUV system was specifically devised for viewing additive sources such as colour monitors (Travis, 1991; Carter and Carter, 1983) and is therefore the best colour system for our purposes (Cowan and Ware, 1985).

Also, the CIELUV system is the most flexible because it is the only system which can incorporate size corrections (Carter, 1996). All colour difference equations are based around the CIEXYZ system which is only defined for colours that occupy 2 degrees of the user's field of view. But our colour perception degrades as the stimulus size decreases (Wyszecki and Fielder, 1971). Silverstein and Merrifield (1985) therefore produced scaling factors for the CIELUV colour space to model the effect of stimulus size on colour perception. We can therefore use their equations to better represent the perceived colour difference for very small stimuli, such as single or small groups of pixels.

3.1.4.3 Implementing the Feature Extraction Stage

Taking into consideration the above discussion, we can envisage a system which locates a visual feature in an image by recursively merging all adjacent pixels (or groups of pixels of the same colour) based upon the perceptual criterion which we have just introduced. That is, whether the group of pixels to be merged is within the prescribed JND threshold of the adjacent pixels already in the feature; using a size-corrected CIELUV colour difference equation (we will discuss the specification of this JND threshold later in Section 3.1.8.3).

Every pixel which is determined to be part of a feature could be copied to a **feature bitmap** in order to isolate the feature from the image. The feature bitmap does not need to hold full colour information for each pixel, only whether a pixel is in the current feature or not. The feature bitmap therefore need only contain binary information (e.g. a monochrome bitmap).

The contrast of the feature can be calculated by finding the ratio of the average luminance of each pixel in the feature, to the average luminance of the pixels immediately surrounding the feature (Peli, 1990).

In summary, a single iteration of the feature extraction stage produces the description of a single visual feature (e.g. a binary feature bitmap containing only those pixels which exist in the feature), and a value of contrast for that feature. This information is then passed onto the second stage for further processing. The above procedure is then repeated until all visual features have been found and processed. (N.B. Appendix B contains a number of examples of the feature extraction process for various images.)

3.1.5 Calculating the Relative Spatial Frequencies

We have now located a 2D visual feature in the image. The next stage is to calculate all of the appropriate relative spatial frequencies which define that feature. These frequencies will be relative because they are only in terms of pixels, i.e. they are a measure of the number of pixels a feature extends over. So the units of spatial frequency at this stage will be cycles per pixel (c/pixel).

3.1.5.1 Developing a Methodology

We have seen from Section 2.3.3.4 (Page 61) that in order to describe a 2D feature in terms of spatial frequency we must incorporate an orientation parameter. Spatial frequency is simply a measure of the size of a stimulus (to be correct, it is inversely proportional to size). Therefore, we effectively want to calculate the largest size of the feature—e.g. the longest contiguous line of pixels—at a number of orientations. Figure 3.2 attempts to illustrate this notion by presenting a feature along with three of its relative spatial frequencies.

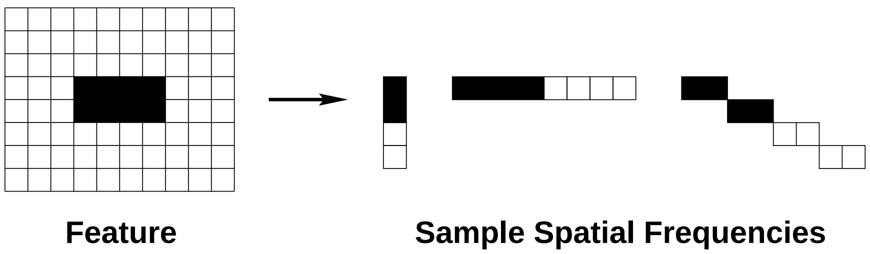


Figure 3.2: *A simple rectangular feature (left) and three of the feature's relative spatial frequencies at various orientations (right). The white pixels in the sample spatial frequencies are required in order to produce a complete contrast cycle in each case.*

If we take the case of the horizontal spatial frequency (0°) in Figure 3.2, then we can use this example to describe the method of calculating the c/pixel values for any feature. The longest horizontal line in the feature is 4 pixels in length, so the relative spatial frequency will be inversely proportional to 4 at that orientation. We can note that in terms of the bars in a square-wave contrast grating, our line of pixels is actually half a contrast cycle: a full cycle has a peak and a trough, i.e. two distinct features. We must therefore apply a scaling factor of 1/2 to our calculation to compensate for this fact. Putting all of this together, we can state that the horizontal relative spatial frequency of the feature in Figure 3.2(left) is: $1/2 \times 1/4 = 1/8$ c/pixels. That is, one complete contrast cycle over 8 pixels. This can be confirmed by looking at the middle spatial frequency in Figure 3.2(right).

From this example, we can develop a general relationship. If we know $l(\theta)$, the length of the longest contiguous line of pixels in a feature at orientation θ, then we can calculate the value of $RSF(\theta)$, the relative spatial frequency of the feature at orientation θ, as follows:

$$RSF(\theta) = \frac{1}{2l(\theta)}. \tag{3.1}$$

3.1.5.2 Implementing the Frequency Extraction Stage

Using Equation 3.1, we can find the relative spatial frequencies in a feature at any orientation. In order to resolve this however, we need to know how to evaluate $l(\theta)$: the length of the longest contiguous line of pixels in the feature at orientation θ. This was a trivial exercise in the example above because the feature was a simple rectangle; but how do we calculate $l(\theta)$ for more complex shaped features? This problem resolves to: how do we find the longest contiguous line of pixels, for a particular orientation, which can occur anywhere in the feature? The technique which the author used to solve this problem is described below.

Rather than attempt to formulate a mathematical solution based upon the geometry of the feature, a direct analysis of the feature bitmap was favoured: for any particular angle, a line of that orientation is notionally scanned through the feature bitmap. At each position, the largest number of contiguous (lit) pixels which exist on that line is recorded. The value of $l(\theta)$ is therefore the largest such result for every line which is passed through the feature. This process is repeated for any number of required orientations.

This method of direct analysis has the convenient benefit of implicitly handling all of the degenerate cases of feature shapes, e.g. concave features, and features with holes (sub-features). Figure 3.3 provides a depiction of this concept in three general cases.

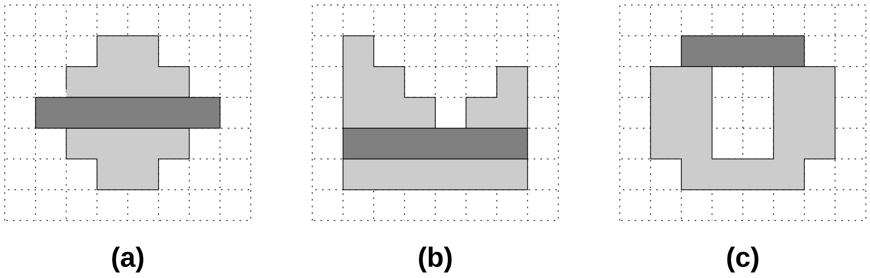


Figure 3.3: *An illustration of the maximum horizontal length of a number of features, showing (a) a convex feature, (b) a concave feature, and (c) a feature with a hole. The bold line of pixels represents the first occurrence of the longest contiguous line of pixels which will ultimately be used to calculate the spatial frequency at that orientation.*

3.1.6 Scaling the Relative Spatial Frequencies

The result of the previous two stages has been to calculate the relative spatial frequency for each feature in an image (at a number of orientations). These values are provided in units of cycles per pixel (c/pixel); however we want them to be available in units of cycles per degree (c/deg) in order to make any perceptual classifications. This naturally requires us to know the horizontal and vertical field of view which the display device occupies. In a head-mounted display system, this is a trivial matter because the FOV information is provided by the manufacturer. However for a standard monitor or projection screen, the FOV must be calculated as a function of the display size and viewing distance. This can be solved by referring to the tan rule for right-angled triangles, e.g.

$$FOV_{\mathrm{horiz}} = 2 \times \tan^{-1}\left(\frac{width/2}{distance}\right), \qquad (3.2)$$

$$FOV_{\mathrm{vert}} = 2 \times \tan^{-1}\left(\frac{height/2}{distance}\right). \qquad (3.3)$$

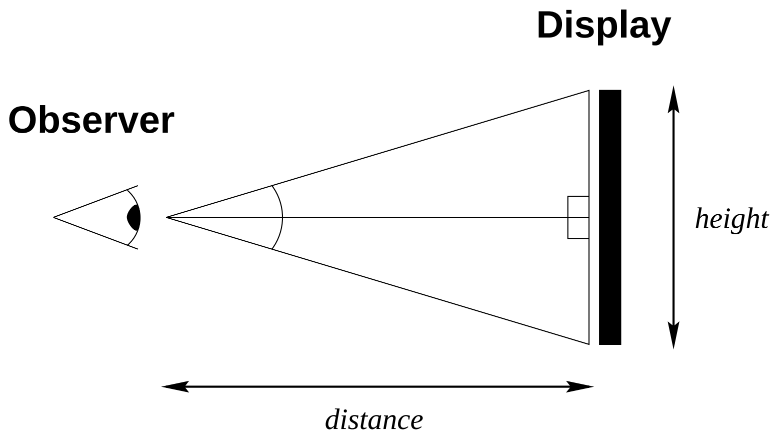


Figure 3.4: *Calculating the field of view of an arbitrary display device.*

From these transforms we can calculate the visual arc subtended by one pixel and hence convert the values of c/pixel into c/deg for all orientations. In the process of performing this, we wish to be able to support any arbitrary rectangular display FOV and any (not necessarily similar) rectangular display resolution. That is, the scaling factor in the vertical direction will not necessarily be the same as the scaling factor in the horizontal direction. In order to accommodate this, we can implement the process by: extracting the horizontal and vertical components of the frequency, scaling these components independently, and then recombining these to give the final result. These three steps are detailed below:

1. Extracting the horizontal and vertical components of the spatial frequency $RSF(\theta)$, at orientation θ, can be achieved by applying the trigonometrical formulae $\cos(\theta) = a/h$ and $\sin(\theta) = o/h$ for a right-angled triangle; where the hypotenuse of the triangle equals $RSF(\theta)$, i.e.

$$C_{\text{horiz}} = RSF(\theta)\cos(\theta), \tag{3.4}$$

$$C_{\text{vert}} = RSF(\theta)\sin(\theta). \tag{3.5}$$

2. In order to scale the value of c/pixel into c/deg, we need to know the resolution of the display device (in pixels) and its FOV (in degrees). Then, we can define the horizontal and vertical scaling factors as:

$$S_{\text{horiz}} = width/FOV_{\text{horiz}}, \tag{3.6}$$

$$S_{\text{vert}} = height/FOV_{\text{vert}}. \tag{3.7}$$

3. Finally, once both the vertical and horizontal components have been independently scaled, we can recombine the two components using Pythagoras' equation to give the resulting absolute spatial frequency, $SF(\theta)$, in units of c/deg:

$$SF(\theta) = \sqrt{(S_{\mathrm{horiz}} C_{\mathrm{horiz}})^2 + (S_{\mathrm{vert}} C_{\mathrm{vert}})^2}. \tag{3.8}$$

One final point which should be noted is the differential in perceived line length which is experienced with relation to orientation. E.g. a horizontal line of 10 pixels will appear shorter than a 45° line of 10 pixels. This can be easily explained by referring once again to Pythagoras' equation: a right-angled triangle with both perpendicular sides equal to 10, will have a hypotenuse of $\sqrt{10^2 + 10^2} = 14.142$, i.e. a 45° line of 10 pixels—which spans 10 pixels horizontally and 10 pixels vertically—will be perceived 1.4 times larger than a horizontal line of 10 pixels. We should therefore make an attempt to scale appropriately the relative spatial frequency value, $RSF(\theta)$, depending upon its orientation, before converting it into an absolute value, $SF(\theta)$. This can be achieved if we assume that a line of x pixels length (at any orientation) will always span either x pixels horizontally or x pixels vertically. This is the case for most line drawing algorithms, e.g. the standard Bresenham's algorithm (Kingslake, 1991). Therefore, we can calculate the value of $RSF'(\theta)$, which incorporates the appropriate compensatory scaling factor for any orientation, $\theta = [0..180^\circ]$, as:

$$RSF'(\theta) = \begin{cases} \sqrt{RSF(\theta)^2 + (RSF(\theta)\tan(\theta))^2}, & \text{when } 0^\circ \leq \theta \leq 45^\circ \\ & \text{or } 180^\circ \leq \theta \leq 135^\circ \\ \sqrt{RSF(\theta)^2 + (RSF(\theta)\tan(90^\circ - \theta))^2}, & \text{when } 45^\circ \leq \theta \leq 135^\circ. \end{cases}$$

3.1.7 What's Wrong with Fourier Analysis?

3.1.7.1 Introduction to Fourier Analysis

The technique of Fourier analysis can be used to decompose an image function into the set of harmonic intensity functions which sum to give the original image (Bracewell, 1965). This transformation is normally represented mathematically as $F(u, v) = \mathcal{F}\{f(x, y)\}$, where $f(x, y)$ represents the spatial domain of the original image and $F(u, v)$ represents the frequency domain of the Fourier transformed result.

The Fourier transform requires a continuous function to operate on; however, there exists a machine-computable method for calculating the Fourier transform of a discrete function which is called the Discrete Fourier Transform (DFT). The formula for the 2D DFT can be defined as follows:

$$F(u, v) \equiv \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}. \tag{3.9}$$

In practice however, computing this function directly is impractical. The most common method is to use the Fast Fourier Transform (FFT) which drastically reduces the complexity of the DFT calculation (Brigham, 1974).

3.1.7.2 Problems with Fourier Analysis

On first inspection, Fourier analysis sounds like a perfect solution to our problem: it takes a 2D source image and returns a frequency domain containing all of the relative spatial frequencies in that image. However, this is not actually what we require.

To illustrate this, Figure 3.5 presents the FFT of two simple images: a contrast grating and a square. In the first instance, the FFT of the contrast grating gives the expected results. There are three visible points. The centre point is the D.C. term and is not relevant to the analysis. The other two points represent a relative spatial frequency at orientations 0° and 180° which correlates with that of the contrast grating. (Note that the Fourier domain is symmetrical about the D.C. term.)

However, when we look at the FFT for the image of a single square, we are presented with a frequency domain which contains values across the entire spectrum of relative spatial frequencies (including ones which cannot possibly exist in the image). Part of the reason for this is that Fourier analysis only decomposes the sinusoidal intensity functions of an image, but the simple square image is effectively a square-wave intensity distribution and so it is rich in harmonics. Therefore, if we apply Fourier analysis to a square-wave intensity function (as might be produced by a flat-shaded feature in an image), then this will instantly produce spatial frequencies over the entire range of frequencies.

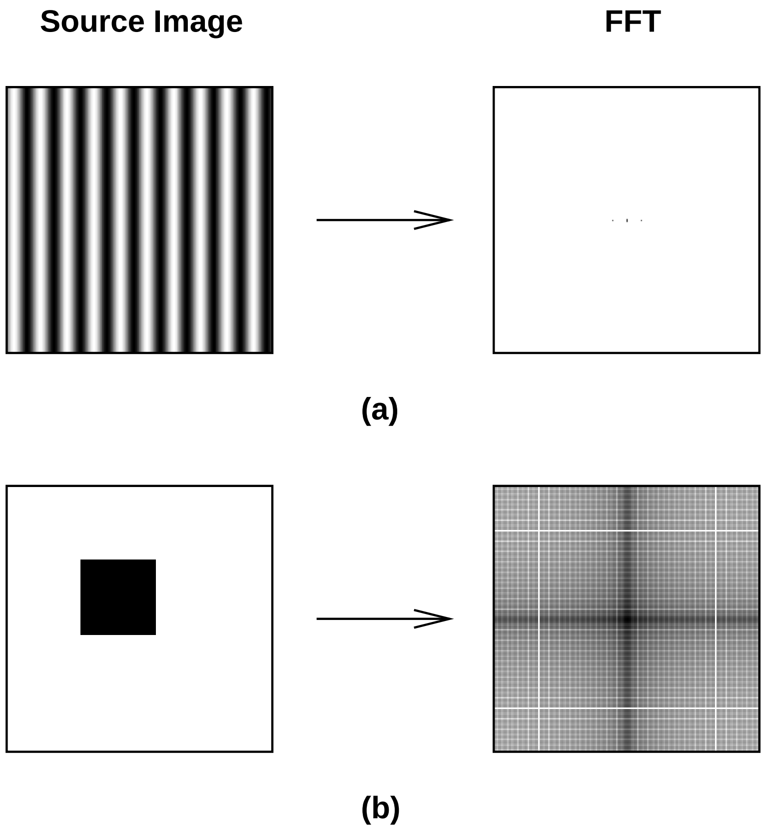


Figure 3.5: *The Fourier transform of two images: (a) an image of a contrast grating, and (b) an image of a square. The figures on the right represent the Fourier transform of the figures on the left, computed using the FFT algorithm.*

Specifically, we can conclude that Fourier analysis is not applicable to our application because of the following confounding factors:

1. Fourier analysis will only reveal harmonic spatial frequencies. Any square-wave variances will introduce substantial artifacts in the frequency domain. These cannot be removed from the frequency domain because, given any arbitrary point, we have no information to discern whether this was produced by a physical feature in the image or by the sine-wave approximation of a square-wave feature. That is, Fourier analysis will only (accurately) extract perfectly harmonic variances in intensity[1].

2. Fourier analysis will return values for all spatial frequencies at every point

[1]There exist other transforms, such as the Walsh Transform (Gonzalez and Woods, 1992), which can locate square-wave intensity components. However, these will then suffer from an inability to locate smooth intensity gradients, i.e. any such features will be recorded as a series of small square-waves instead of a more appropriate sine-wave coding.

in the image. So for example, if we take the situation with the image of the square: the 0° (horizontal) spatial frequency at the centre of the square will be relatively low, but the 45° frequency towards the top-left corner will be very high. That is, any 2D feature with a non-smooth boundary edge will always return frequency values across the entire spectrum.

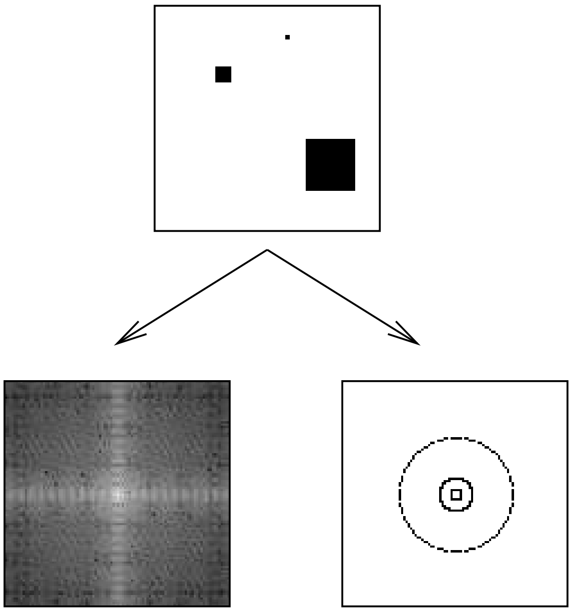
The image segmentation algorithm circumvents these problems because it only extracts the *fundamental* spatial frequency (i.e. the lowest frequency) for every visual feature, at each desired orientation. The rationale behind only extracting the fundamental frequency is that if the lowest frequency is not detectable by the observer, then none of the higher frequencies will be detectable either—so we only need to record the fundamental frequency for a feature to decide whether it is visible under any specific viewing condition. This concept cannot be easily encoded into a Fourier process.

3.1.7.3 Spatial versus Frequency Domain Analysis

Given the above discussion, we can see that the image segmentation approach offers a number of advantages over the Fourier analysis method. These can be summarised as follows:

1. The image segmentation system can be applied to images that contain both flat and smooth shaded features (including antialiased and texture mapped features).
2. The corresponding frequency domain is noise-free with no high frequency artifacts. It is therefore much more amenable to accurate analysis.
3. The results are restricted to the fundamental frequency of each visual feature; instead of overloading us by reporting every spatial frequency in the image.
4. We have access to information about the physical position of a feature in the image (e.g. the position of a feature could be defined as the centre point of its bounding square). This is not available under the Fourier method.

The impact of these advantages can be easily ascertained by inspecting Figure 3.6. This presents a simple image with three squares of various sizes, and then displays the FFT of that image alongside the results of the image segmentation algorithm (plotted in Fourier space). As can be observed, the FFT result is highly noisy and it is unlikely that this could be of any use as an accurate measure of visual detail. On the other hand, the image segmentation process shows a concise and accurate result: it contains three discernible circles which represent the fundamental spatial frequencies of the three squares at each orientation.



(a) **(b)**

Figure 3.6: *A comparison of (a) the Fourier based, and (b) the image segmentation based approaches for locating the relative spatial frequencies within a sample image. (Both results are plotted in the Fourier domain.)*

It is certainly possible to investigate more sophisticated Fourier techniques, however it is the author's belief that this would be attempting to shoehorn a solution which patently does not fit in this situation. Although it is true that Fourier techniques have helped vision scientists to understand how the brain processes simple harmonic stimuli such as contrast gratings, recent theories of visual processing indicate that spatial detail is analysed on a local basis and that the visual system does not operate under Fourier principles (Daugman, 1984; Ware and Knight, 1995).

3.1.8 Discussion of the Measure for Perceived Detail

3.1.8.1 An Assessment of the Adopted Approach

We have presented a technique to extract all of the important spatial frequencies from a full-colour computer-generated image. This works by extracting features from an image, computing their relative size, and then scaling these into units of c/deg once the display FOV is known.

Of these three stages, the feature extraction process is the most subjective; with the other two stages being relatively well defined. Unfortunately, the field of visual perception can offer us little help to resolve the problem of defining a single 'feature' in a complex image. We have therefore devised a method which effectively locates the edges of features based upon whether there is a perceived colour differential between adjacent groups of pixels. This has some basis in the neural anatomy of the visual system (e.g. receptive fields are designed to located edges in the retinal image); however, it should be noted that there is no formal basis for such an approach[2].

As such, this model is certainly not proposed as a general model for how we perceive detail. However, it does perform well for the subset of computer-generated images, i.e. the model works best when there is no noise in the image and when surfaces exhibit smooth variations in colour. Both of these conditions are prevalent in computer-generated imagery, but less common in the real world.

One question which the discerning reader might be asking themself at this stage is why do we attempt to extract features based upon chromatic flux when we have already stated that we will use achromatic threshold data to model the visibility of a feature? The answer to this is that the purpose of the feature extraction process is to capture the *suprathreshold* perception of a scene. We can then apply the contrast sensitivity model to discover which of these features are below threshold (not visible) for any given viewing condition. If we segment the image based only upon the luminance information then we might not obtain an accurate perceptual segmentation for isoluminant colours. For example,

[2] As an interesting excursus, we can draw some qualitative comparisons between our approach and the Gestalt principles of organisation, e.g. we attempt to merge neighbouring parts of an image together (proximity), pixels are merged if their colour is sufficiently close to the rest of the feature (similarity), and features are solid 2D collections of pixels (closure).

consider a berry bush with green foliage and bright red berries. When we look at this scene in colour we are immediately aware of the berries as discrete entities (features). However, if we only look at the luminance information for this scene then we hardly notice the berries (if at all) because they have a similar intensity to the surrounding foliage. Mullen and Kingdom (1991) present a similar example and state that: '*any chromatic differences* [...] *would therefore be particularly useful for perceptual segregation*'.

3.1.8.2 An Evaluation of Colour Difference Formulae

Unfortunately, there is currently no accurate or principled measure for perceived colour difference, e.g. we have already noted that even the CIE's standard colour spaces (CIELUV and CIELAB) can exhibit a ~6:1 error. In fact, Hardin (1988) suggests that it is not possible to find a single scheme to accurately represent all of the basic appearance attributes of surface colours.

The problem is that individuals' colour perception seem to vary quite widely. Wyszecki and Fielder (1971) report that agreement of results between any pair of observers is poor, and that even the ability of a single observer to match the same colours varies substantially over time. They state that averaging the data of several observers does not yield a representative result for an intermediate observer.

In addition to this subjective problem, there are a number of physical factors which can affect the appearance of colours on a computer graphics display (MacDonald *et al.*, 1990; Meyer and Greenberg, 1980). Some of these are beyond our simple control (such as the ambient light in the user's environment). However, some of the more accountable factors include:

1. **Stimulus Size** : colour differences vary in relation to the size of the two colours, but the CIE colour spaces are only defined for colours occupying 2 deg of visual arc. However, we have already noted that size correction formulae exist for the CIELUV colour difference equation (see Carter, 1989, for a review).

2. **Surrounding Colour(s)** : the brightness of a colour can appear different for different surrounding colours. This effect is know as **simultaneous contrast**. Phillips (1986) reports that this can induce perceived colour

difference variations in the order of 10–20%. However, when compared with the variability of a typical observer, this error does not appear to be significant (Phillips, 1986). More critically, this effect is of course the result of our human perceptual processing (the wavelength of light reflected by an object does not physically change due to juxtaposition with another object). This processing occurs at a later stage than the early visual detection stage that we are concerned with. We therefore do not need to compensate for this phenomenon in our application.

3. **Display Factors** : a number of monitor characteristics can affect the perceived colour of a stimulus. For example, a monitor's black level (or brightness) and its gamma value (Poynton, 1993). Care should therefore be taken to ensure that the display's brightness control is set so that dark elements are reproduced correctly, and that all colours are gamma corrected (see Reddy, 1996a).

It is therefore evident that any colour difference formulae which we employ will not give us a consistent and reliable estimate for a user's ability to discern between two colours. However these are the best measures that we currently have at our disposal.

3.1.8.3 Specifying the JND Threshold for Feature Extraction

We have left until last the question of how to specify the JND threshold for the feature extraction process. That is, what is the value of CIELUV colour difference (ΔE^*_{uv}) below which the user cannot perceive any change between two colours? This topic is not tackled to any extent by the colour perception literature; although a few references talk about CIELUV units in the order of 3–10 (Barten, 1990). We therefore decided to try and evaluate this threshold empirically.

An experiment was devised in which an observer tried to match two different colours so that they could perceive no difference. The ΔE^*_{uv} value at this point was noted in each case. Appendix A presents the results of this experiment. From these data we can observe a vast variability in the ΔE^*_{uv} results (ranging from $\sim$2 up to $\sim$30 CIELUV units).

Given the inherent problems of specifying colour perception which we have just described, the final decision of the threshold figure is quite arbitrary. We therefore decided to take the mean ΔE^*_{uv} value for all of the experiment trials. This evaluated to 10. I.e. within our measure for perceived detail, two pixels were considered equal if their size-corrected CIELUV colour difference fell below 10.

3.1.8.4 Example Spatial Frequency Analysis

To help illustrate the technique that we have just described, Figure 3.7 presents an example analysis with three different levels of detail for a model of a die. Next to each LOD is a graph of the relevant spatial frequencies in that image. The abscissa represents increasing spatial frequency (c/pixel), i.e. regions of high detail are coded to the right of the graph. The ordinate denotes the number of features in the image with a particular spatial frequency.

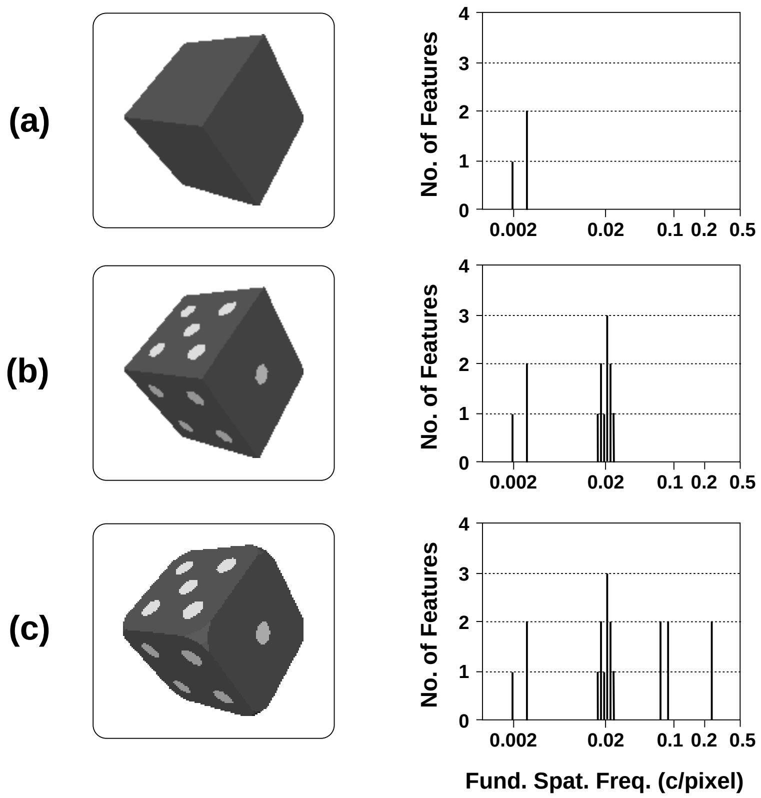


Figure 3.7: *An example spatial frequency analysis of three different LOD models. In each graph, the abscissa represents increasing spatial frequency (c/pixel).*

From these results we can note that there appear to be three major groupings of spatial frequency. In the case of the lowest LOD, Figure 3.7(a), we see that there are only three low spatial frequencies. These will obviously represent the three visible faces of the die; which are the only features at that level. In the medium LOD, Figure 3.7(b), we can see that we still have these three low spatial frequencies, but we now also have a batch of higher spatial frequencies. These will represent the spots on the die which have been introduced at that level. Finally, in the highest LOD, Figure 3.7(c), we can see the same trends as in the other two instances, but we also have some even higher spatial frequencies. These will represent the added detail induced by the curvature of the die at its edges.

From this analysis we can see that we have a high LOD which contains a number of high spatial frequencies, and that for each lower LOD we have fewer of these high spatial frequencies (less high detail). Therefore, if this die was presented to a user in such a situation where they could only perceive frequencies below 0.01 c/pixel, then we could select the lowest LOD model—Figure 3.7(c)—and the user would not be able to perceive any change.

3.2 Modeling Contrast Sensitivity

Now that we have formulated a metric to assess the perceptual content of an image in terms of spatial frequency, we require a means to predict how much detail a user can perceive in an image at various velocities and eccentricities.

We have seen from the previous chapter that the CSF defines the efficacy of the human visual system to resolve a stimulus based upon its size and contrast. However, these curves are produced by interpolating tabulated values obtained through empirical studies. In order to usefully incorporate these results into a computer graphics system we need to find a mathematical model to describe the shape of the human CSF.

Kelly (1975) embarked upon just such an activity. He developed a conceptual model to describe the spatial frequency characteristics of retinal receptive fields at high illuminance levels ($>$ 900 **trolands**), and showed that this can be used to model the sine-wave sensitivity of the visual system. Kelly's abstract model for contrast sensitivity is defined as:

$$F(\alpha) = \alpha^2 \exp(-\alpha), \tag{3.10}$$

where α represents spatial frequency (c/deg). (Recall that contrast sensitivity is defined as the reciprocal of threshold contrast, i.e. threshold contrast can be modeled by $1/F(\alpha)$.)

With suitable scaling factors, this general equation can be used to model the shape of the CSF under various viewing conditions. For our purposes, we wish to take into consideration the effects of velocity and eccentricity. We will therefore investigate how these variables can be incorporated into Equation 3.10.

3.2.1 Incorporating Velocity into the Model

The surface which is produced by mapping the CSF for a range of velocities is called the **spatiotemporal threshold surface**. This has been investigated by a number of vision researchers over the years. The most notable of these is D. H. Kelly, who was engaged in the study of the spatiotemporal surface for over 20 years. Our model will be based upon his findings. It is worth noting however that Burr and Ross (1982) conducted similar experiments to those of Kelly, and that their principal results correlate almost exactly with his.

Kelly (1979) made extensive studies of the spatiotemporal surface under conditions of stabilised vision[3]. From his data, he noted that the shape of the CSF remains essentially constant for all velocities above 0.1 deg/s, and only undergoes translation with increased velocity. He subsequently extended Equation 3.10 to model the spatiotemporal threshold surface (for velocities above 0.1 deg/s), by introducing two scale factors: k (the height of the CSF), and $\alpha_{\max}$ (the peak frequency of the CSF). His equation can be presented as follows:

$$G(\alpha, v) = kv\alpha^2 \exp(-2\alpha/\alpha_{\max}), \tag{3.11}$$

where,

$$k = 6.1 + 7.3|\log_{10}(v/3)|^3, \tag{3.12}$$

[3]A non-contact method of accurately stabilising the retinal image was used. This enabled measurements to be taken without the artifacts introduced by uncontrolled eye movements.

$$\alpha_{\max} = 45.9/(v + 2). \quad (3.13)$$

Where v represents velocity measured in units of deg/s, and α represents spatial frequency in units of c/deg. Equations 3.11–3.13 can be combined into a single expression to give:

$$G(\alpha, v) = [6.1 + 7.3|\log_{10}(v/3)|^3]v\alpha^2 \exp[-2\alpha(v + 2)/45.9]. \quad (3.14)$$

However, the author was unable to reproduce the empirical data which Kelly (and others) presents using this formula. Therefore, a modified version of Equation 3.14 was developed to more accurately model the available data. This was determined empirically with computer-aided techniques. The final solution was obtained by effectively altering the weighting of the k and $\alpha_{\max}$ components (with weightings of 41.0 and 2.75, respectively), and converting the base of the exponential term to base 10. The resulting equation can be represented as follows:

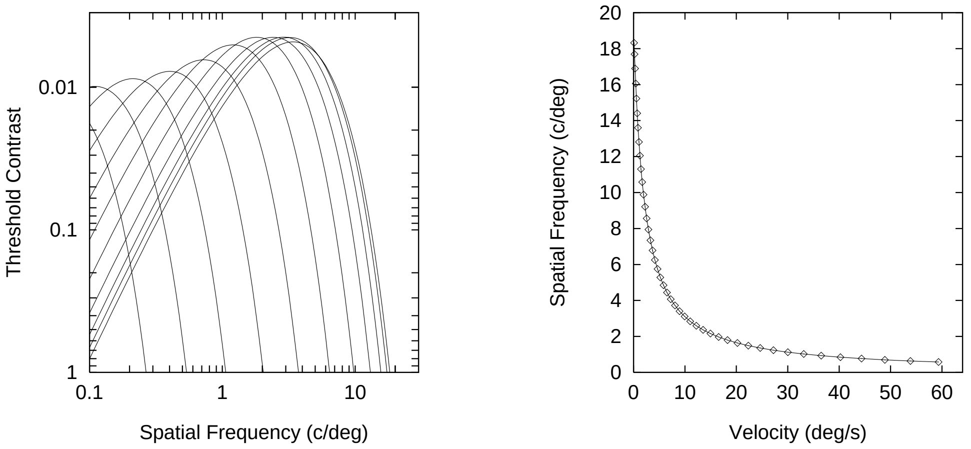
$$G(\alpha, v) = [250.1 + 299.3|\log_{10}(v/3)|^3]v\alpha^2 10^{-5.5\alpha(v+2)/45.9}. \quad (3.15)$$

This equation is plotted in Figure 3.8(a) for a number of velocities. From this we can observe that the effect of velocity on the CSF is to push the curve further towards the ordinate for higher velocities. This enforces what we have already stated back in Section 1.4.4.3: that we can see less high detail with increasing velocity. Figure 3.8(b) attempts to portray this relationship more clearly by plotting the highest visible spatial frequency for a range of velocities.

3.2.2 Incorporating Eccentricity into the Model

Contrast sensitivity declines with increasing eccentricity. However the shape of the spatiotemporal surface is consistent across the visual field (Virsu *et al.*, 1982; Koenderink *et al.*, 1978b; Kelly, 1984). This would lead us to believe that we can predict the contrast sensitivity for any region of the retina by simply scaling the foveal response with a factor based upon eccentricity.

Rovamo and Virsu (1979) confirmed this when they showed that visual acuity can be accurately predicted for any eccentricity by applying a constant scaling



(a) **(b)**

Figure 3.8: *(a) Contrast sensitivity functions for velocities of 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, and 128 deg/s (from right to left); calculated using Equation 3.15. (b) The highest visible spatial frequency (visual acuity) for a range of velocities, i.e. the upper point of intersection with the abscissa for each of the curves in (a). The data points represent calculated intersection points, with the curve illustrating the interpolated relationship.*

factor, referred to as the **cortical magnification factor** (M), first introduced by Daniel and Whitteridge (1961)[4]. Therefore, in order to incorporate eccentricity into our model of spatiotemporal contrast sensitivity, we simply need to apply this cortical magnification factor to Equation 3.15.

The eye's peripheral sensitivity is not circular symmetric (e.g. Regan and Beverley, 1983). For example, there are marked asymmetries between the **nasal** and **temporal** retina beyond 20 degrees (Sutter and Tran, 1991). Taking this into consideration, Rovamo and Virsu produced four equations to characterise M for each principal half-meridian of the retina. These are replicated below, and plotted in Figure 3.9(a).

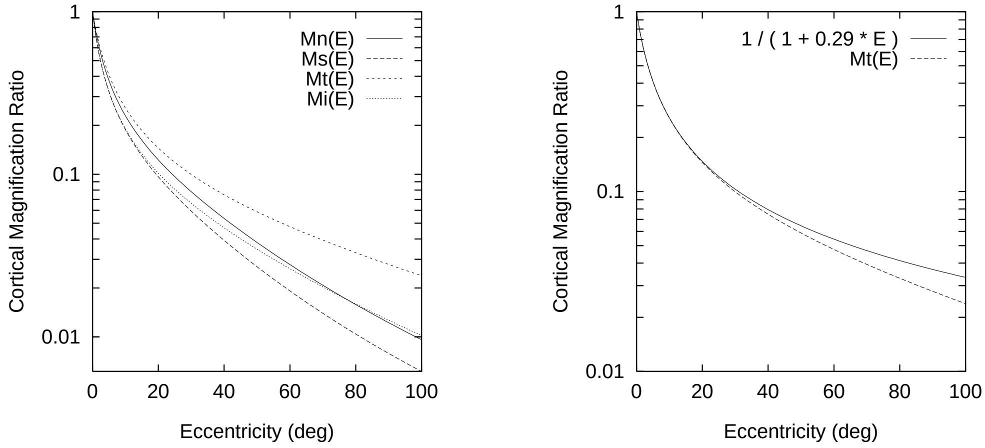
$$\text{Nasal:} \quad M_N = M_0/(1 + 0.33E + 0.00007E^3), \quad 0 \le E \le 60 \text{ deg.} \tag{3.16}$$

$$\text{Superior:} \quad M_S = M_0/(1 + 0.42E + 0.00012E^3), \quad 0 \le E \le 45 \text{ deg.} \tag{3.17}$$

$$\text{Temporal:} \quad M_T = M_0/(1 + 0.29E + 0.000012E^3), \quad 0 \le E \le 80 \text{ deg.} \tag{3.18}$$

$$\text{Inferior:} \quad M_I = M_0/(1 + 0.42E + 0.000055E^3), \quad 0 \le E \le 60 \text{ deg.} \tag{3.19}$$

[4]Where M^2 is directly proportional to the density of receptive fields of retinal ganglion cells (Drasdo, 1977).



(a) **(b)**

Figure 3.9: *(a) A graph of Equations 3.16–3.19, which define the cortical magnification factor, M, for each cardinal half-meridian of the retina. (b) A comparison of Equation 3.18 with the cubic term included (lower curve) and with the cubic term ignored (upper curve).*

Where M_0 is the value of magnification for the most central point in the fovea; which we can simply instantiate as $M_0 = 1$. An intelligent system would take into account the relevant region of the retina which is being considered and apply the appropriate value of M for that region. Alternatively, to simplify the relationship, we could exclusively use the most sensitive region's M, with the knowledge that the other regions will not exceed this sensitivity; i.e. Equation 3.18 (M_T).

It would be reasonable to ignore the cubic term in Equation 3.18. This only becomes significant at large eccentricities; and even when $E = 100$ deg, there would only be a $\sim$1% error. This simplification was adopted by Watson (1983), Kelly (1984), and Tyler (1985) in their respective models, among others. We can therefore define the cortical magnification factor for our purposes as:

$$M = M_0/(1 + 0.29E). \tag{3.20}$$

We can subsequently incorporate this equation into our model for contrast sensitivity as follows:

$$\begin{aligned} H(\alpha, v, E) &= G(\alpha, v) \times M, \\ &= G(\alpha, v)/(1 + 0.29E). \end{aligned} \tag{3.21}$$

3.2.3 Modeling Visual Acuity

We have seen from Section 2.3.4.1 (Page 63) that it is possible to describe the visual acuity of an observer in terms of spatial frequency, given their contrast sensitivity. We will therefore illustrate how this may be achieved for our model of contrast sensitivity.

In essence, we only wish to calculate the upper point where the CSF intersects the abscissa. Beyond this point, detail is not visible; irrespective of the level of contrast applied. Figure 3.8(b) illustrates this notion by plotting the velocity of a stimulus against the highest spatial frequency which is visible at that velocity (where $E = 0$ deg). From this figure we can observe that our visual acuity drops precipitously as a stimulus moves with greater velocity, up to about 10 deg/s, and then begins to level off asymptotically.

The point at which the CSF meets the abscissa is defined as a contrast sensitivity of one, i.e. $H(\alpha, v, E) = 1$. The solution to calculating the highest detail which a user can perceive for any velocity and eccentricity can therefore be written as:

$$H(\alpha, v, E) - 1 = 0. \tag{3.22}$$

We then need to solve this equation in terms of α in order to discover the spatial frequency at which $H(\alpha, v, E) = 1$, taking the highest root as the threshold size. An analytical solution to this problem would be overtly complicated; requiring the computation of Lambert's W function (Corless *et al.*, 1993), or equivalent, to resolve the combination $\alpha^2 \exp(\alpha)$ which arises. A more tractable solution can be found by using an iterative method such as an interval halving algorithm or a Newton-Raphson technique.

3.2.4 Discussion of the Visibility Model

Equation 3.21 provides us with a means of estimating the contrast sensitivity of a human observer to a stimulus of any size (α c/deg), moving at a particular velocity ($v >$ 0.1 deg/s), and located at any arbitrary eccentricity (E deg). We have also shown how we can describe a subject's visual acuity using the same parameters.

The velocity part of our model has been derived from the work of Kelly (1979).

His formula was produced analytically by attempting to find a general equation to model the results from various drifting contrast grating tests. These experiments were performed under stabilised vision in order to find the vision system's ideal spatiotemporal response. However, we do not view objects in the real world under stabilised conditions (i.e. our eyes exhibit small uncontrollable movements which can introduce motion artifacts). There is also evidence to suggest that, to a certain extent, our ability to perceive detail in moving objects is a skill which can be improved with practice (e.g. Sekuler and Blake, 1994; Murphy, 1978). However, this is probably due to an improved ability to fixate upon a moving target (thus reducing its angular velocity) rather than an adaption of our visual system proper. In addition, Kelly's (1979) work was concerned with responses to periodic sinusoidal gratings. We are of course primarily concerned with the visibility of aperiodic targets and so we may find that Kelly's threshold surface will underestimate the actual threshold surface for our application. It is therefore likely that our velocity model may not accurately model our pragmatic ability to perceive spatial detail under varying velocity. However, the model is as accurate as contemporary visual perception can allow.

It is worth noting that our references to the term velocity have so far implied translational velocity (the angular velocity of a stimulus across the retina). However there is also the possibility of rotational velocity (the angular velocity at which a 3D stimulus rotates on its axes). The field of visual perception contributes no significant knowledge to help us deal with this situation. This is understandable because even today vision scientists are still concerned with simple 1D or 2D harmonic images. However, presumedly, the visibility of a rotating stimulus could be determined if we can resolve its rotational velocity into a translational velocity across the retina.

Our contrast sensitivity model also accounts for the eccentricity of a target by applying the work of Rovamo and Virsu (1979). Their formulae for cortical magnification were based upon published data for the density of ganglion cell receptive fields across the retina. We know that visual acuity varies linearly with M across the retina, and that this is due to a measurable increase in the smallest receptive field size of cells towards the periphery, i.e. this constitutes a fundamental and physical limit to our visual resolution. We would therefore expect that the cortical magnification factor which we have incorporated into our model should provide an acceptable estimate of a user's peripheral sensitivity to spatial frequency.

One factor which is notably absent from our model is the ambient illumination level. We know that our ability to perceive detail varies with respect to the background illumination; with our resolving ability degrading in darker surrounds (Kelly, 1975). However, Kelly's (1979) model was formulated for high background illumination. This is therefore a worst-case model, and so implicitly handles the case of low background illumination. We therefore do not need to consider the effects of ambient illumination in our contrast sensitivity model.

3.3 Perceptually-Driven LOD Generation

With a mechanism to compute spatial frequencies in a computer-generated image, and a computational model for contrast sensitivity, we have now taken care of all the vision related problems. The only other factor which we have yet to address is how to design a polygon simplification algorithm which supports and complements a perceptually-based LOD system. This will be the subject of the current, and final, section in this chapter.

3.3.1 Formulating an LOD Generation Framework

Subsequent to our review of the polygon simplification literature in Section 2.2, we formulated three principal criteria for the use of polygon simplification techniques in a perceptually-based LOD system. To recapitulate, these were: 1) the effect of the simplification should be perceptually predictable, 2) the surface properties of the model should be taken into consideration, and 3) a topology simplifying algorithm should be advocated.

With regards to the last criterion, we do not want to restrict ourselves unnecessarily to any one specific simplification algorithm, because every algorithm offers different advantages and disadvantages. Indeed, as Luebke (1996) notes, it is unlikely that we will find a perfect all-encompassing simplification scheme. We therefore desire a general framework which addresses the first two criteria, but which can support a number of techniques for polygon simplification.

We are already aware that our perception of visual detail is based upon the relative size of a stimulus. Therefore, in order to simplify a model in a perceptually predictable manner, we must be able to guarantee that the degeneration is restricted to a certain scale. The GENIE system—reported by Kemeny (1993) and described in Section 2.2.2.6—achieves this by placing a 2D grid over a projection of the model and then reducing the representation so that only one polygon occupies any one grid cell. This effectively removes all detail below any prescribed scale. However, the technique is limited to two dimensions, whilst we require a general 3D technique.

By extrapolating the GENIE concept, we can suggest a framework for restricting the extent of a simplification process in three dimensions: i.e. instead of using a 2D grid on a projection of the model, we can use a 3D grid on the model itself. Thus we can segment a model into a number of **sub-volumes** and locally perform a simplification algorithm on only those polygons which exist within any particular sub-volume.

By altering the size of the sub-volumes we gain the ability to vary the scale of reduction; e.g. a fine grid mesh will remove only very small features, whereas a coarse grid will remove larger features. The overall effect of this mechanism is to ensure that no changes are made to the model above a certain threshold size. Therefore, if we choose a grid volume which projects to a size smaller than that of a single pixel for a particular viewing distance, then we gain the ability to automatically remove any detail which will not be displayed on the output device.

3.3.2 Describing an Implementation of the Framework

We will describe here how our sub-volume based framework for polygon simplification can be implemented. Figure 3.10 below presents an abstract top-level algorithm to illustrate the general structure and functionality of the framework; whilst the following sections will take each of the major steps in turn and discuss these in greater detail.

3.3.2.1 Geometric Normalisation (Step 1)

The initial normalisation process is an optional preprocessing stage which attempts to provide the geometric information in a standardised and consistent

Step 1:	`Normalise the geometrical data`
Step 2:	`Segment the model into sub-volumes`
Step 3:	`For each sub-volume in the model:`
Step 3.1:	`    Find all polygons entirely within the sub-volume`
Step 3.2:	`    Apply a simplification algorithm to only these polygons`
Step 3.3:	`EndFor`
Step 4:	`Optimise the geometrical data`

Figure 3.10: *Top-level algorithm describing the operation of the sub-volume based polygon simplification framework.*

form. We have noted in the previous chapter that most simplification algorithms only accept models in a certain format (e.g. triangular meshes with no coincident triangles). The purpose of this step is therefore to make our algorithm more accessible by attempting to convert arbitrary polygonal data into a format which most simplification algorithms can utilise. This includes actions such as: triangulating arbitrary polygons; removing multiply-defined vertices, edges, and polygons; removing coincident polygons; etc.

3.3.2.2 Sub-volume Segmentation (Step 2)

The simplest way to segment a model into a number of sub-volumes is to find the bounding box of the object and then choose a granularity with which to divide this volume into smaller rectangular volumes. Figure 3.11 illustrates this concept by presenting a model with an example subdivision grid overlayed. We should note however that the sub-volume need not be rectangular in shape, e.g. a spherical volume could easily be used instead.

3.3.2.3 Sub-volume Restriction (Step 3.1)

For each sub-volume, all polygons which are *entirely within* that volume are located and passed onto the next stage for simplification. This ensures that only polygons whose extents are definitely smaller than the threshold volume are considered for reduction (i.e. polygons which constitute below-threshold detail). This task could be implemented efficiently using an octree data structure to organise the 3D model.

One limitation of this approach is evident: any polygons which lie on the bound-

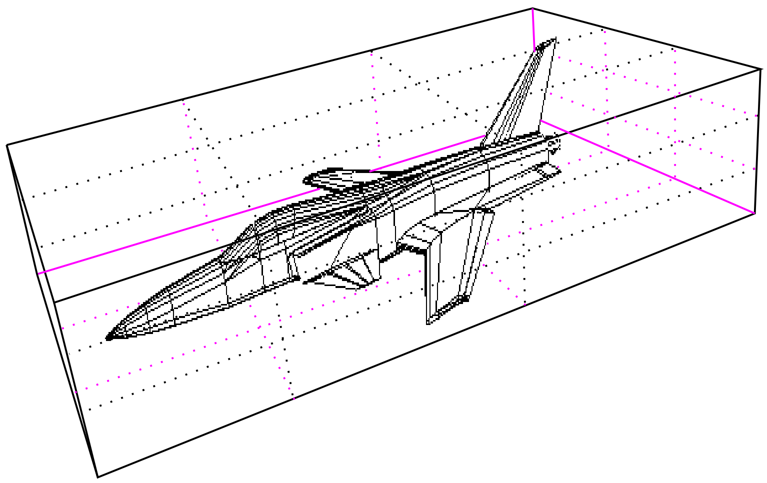


Figure 3.11: *An example model showing a sample sub-volume segmentation. The contents of each sub-volume is considered independently for simplification.*

ary of a sub-volume will never be considered for reduction (because they are not entirely within any sub-volume). The solution to this problem is to overlap successive sub-volumes. As a simple example, if the dimensions of the sub-volume are w, h, and d respectively, then possible positional offsets for a sub-volume can be given by: $(w \times l/2, h \times m/2, d \times n/2)$, where $l, m, n \in [0, 1, 2 \ldots]$.

3.3.2.4 Local Simplification (Step 3.1)

The simplification phase can be implemented using any of a number of polygon simplification algorithms, such as those presented in the previous chapter. For example, Reddy (1996b) presents an implementation of the sub-volume framework using an edge degeneration scheme. This proceeds by trying to collapse certain vertices so that they become co-linear with two of their neighbouring vertices. A reconstruction phase then attempts to rebuild the model in an optimal fashion, merging the resulting pairs of co-linear edges into a single edge.

The sub-volume restriction phase provides us with the assurance that the available polygon subset is entirely contained within the current sub-volume; and so the effects of the degeneration can be constrained to a predefined spatial boundary. However, it is likely that a polygon may contain a vertex which is common to a polygon not entirely within the current sub-volume (as illustrated in Figure 3.12). In this situation, if we alter the position of such a vertex, then we must consequently alter the representation of a polygon which is out with the sub-volume. This side-effect cannot be tolerated if we require that no visual changes are to be made to the model above the sub-division threshold.

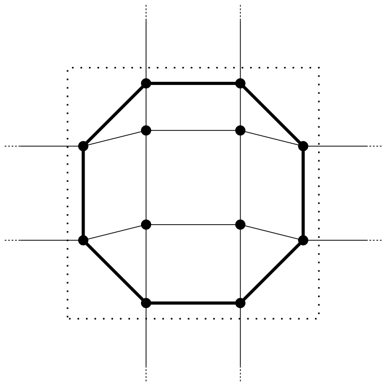


Figure 3.12: *A simple polygon mesh showing the extent of a sub-volume restriction (the dotted square). The bold demarcation illustrates the boundary edges within the sub-volume. Vertices on this boundary should not be considered for simplification.*

To enforce this requirement, a proviso must be added that no degeneration is attempted on a vertex which is shared by a polygon existing out with the current sub-volume (e.g. a vertex which lies on the boundary edge of the sub-volume).

3.3.2.5 Final Optimisation (Step 4)

The last step provides an optional postprocessing stage which attempts to optimise the final polygon mesh. For example, this stage could remove any unused vertices from the model's description, or merge any resultant adjacent, co-planar polygons with identical surface characteristics.

3.3.3 Dealing with Surface Properties

We have now addressed two of our three criteria, but we have yet to consider the effect of surface properties on the simplification. Most simplification algorithms attempt to repeatedly replace a set of adjacent polygons with a smaller set of adjacent polygons that retain the general characteristics of the original grouping. We therefore have to resolve how to define the surface characteristics of any new group of polygons (e.g. their colour, texture, etc.).

If we consider the function of the human visual system once more: our vision system has an amazing ability to integrate over features which are below threshold. For example, if we have a fine mesh of black and white squares and display this at a distance such that it is below the threshold of vision, then we can no longer resolve discrete black and white regions. Instead we perceive a single wash of colour which exhibits the average contrast of the two component colours (Livingstone, 1988). This phenomenon is used to great effect in the newspaper industry where halftoned patterns can be used to portray a wide range of greyscale levels with only a single black pigment on white paper.

To apply this principle to our problem, we can state that the colour of a new polygon, or group of polygons, should be based upon the average colour of all the previous component polygons, i.e. the RGB values for each polygon are averaged. This results in an averaging of luminance (achromatic) as well as colour (chromatic) information, due to the linear relation between them. However, the case is not quite that simple because the relative sizes of the different colour stimuli will affect the perceived result. For example, if the black squares in our mesh were slightly larger than the white squares, then the below-threshold wash would appear darker than when the squares were equally sized. Therefore, the colour of a new polygon grouping is found by an *area-weighted averaging* of the RGB colours for all of the polygons which previously occupied the same volume.

The principles for dealing with texture mapped polygons are exactly the same. In fact, we can think of a single texture mapped polygon as fulfilling the same visual function as a collection of tiny, individually coloured polygons. Stated this way, we can suggest that a good replacement for a texture mapped polygon would be to use an untextured polygon whose colour is found by averaging all of the pixels in the texture map image. This also has the desirable side-effect of replacing textured polygons with less complex primitives. Texture mapped polygons normally incur a greater computational and memory overhead than simple flat or smooth shaded polygons. Therefore, removing these primitives should improve the rate at which the model is rendered, hence adding another facet to the simplification algorithm.

3.3.4 Discussion of Perceptually-Driven LOD Generation

This section has presented a general framework for performing polygon simplification which could be implemented using a range of techniques. The framework enables us to predict the perceptual side-effects of a simplification by restricting the spatial extent of a degeneration.

We introduced one constraint on the simplification algorithm to ensure perceptual predictability (that vertices shared by polygons not entirely within a sub-volume should not be affected). We also suggested an enhancement to enable an algorithm to handle the degeneration of surface properties as well as geometric information (i.e. an area-weighted averaging of the previous surface properties over the extent of the local change).

The result of this development is that we can now accurately predict the screen size at which a model can be selected without introducing any visual artifacts. In many cases this may be an overestimate (e.g. if the largest change is well below the chosen sub-volume size), but the important point is that it should never be an underestimate. It is worth reminding the reader that we will not use this method (the projected size of a sub-volume) in order to define our LOD thresholds; we will of course be using our more general perceptual criteria, i.e. our measure for spatial frequency and our contrast sensitivity model. However, we will see in the next chapter that being able to guarantee that a reduction is restricted to a certain scale will greatly aid and simplify the implementation of a perceptually modulated LOD system.

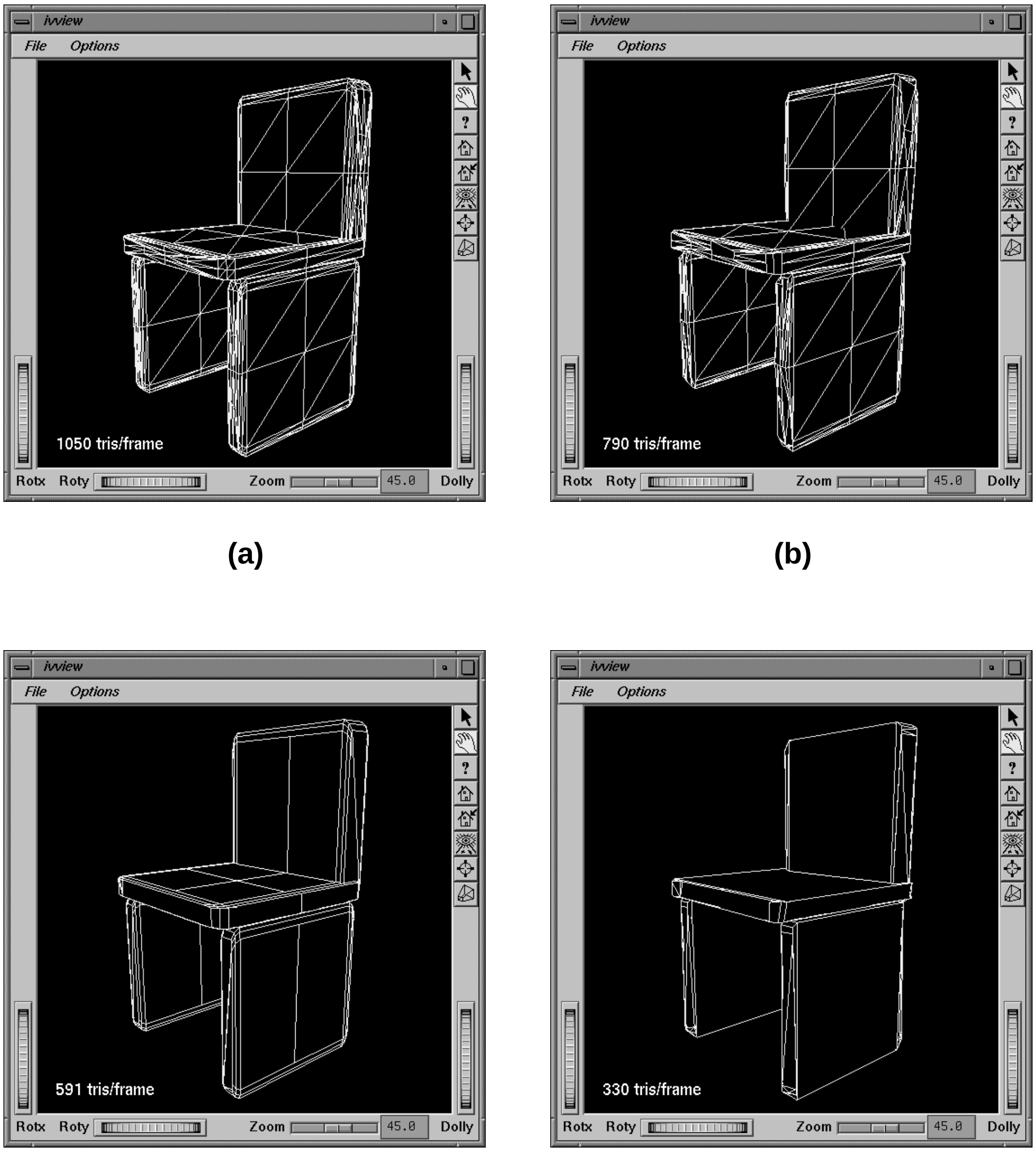
To illustrate the above concepts in practical terms, we present results from Reddy's (1996b) implementation of the framework. Figures 3.13(b)–(d) present the reduced versions of Figure 3.13(a), using sub-volume sizes of one sixty-fourth, one eighth, and all of the object's bounding volume, respectively (i.e. the widths of each sub-volume are one quarter, one half, and equal to the width of the bounding volume, respectively). This implies that the model in Figure 3.13(b) can be used when one sixty-fourth of the object's bounding volume projects to less than a single pixel on screen, and similarly for Figures 3.13(c) and (d).

Table 3.1 below reports the complexity of each model in terms of the number of polygons and vertices which it contains and the average frame rate which was achieved when it was rendered to the screen (using Open Inventor V2.0 on an SGI RealityStation)[5]. The absolute values of update rate are not important, because these will vary across platforms and graphics packages, but their relative magnitudes provide some indication towards the performance benefits which can be accrued when employing each degraded model.

Model	**Polygons**	**Vertices**	**Update Rate**
Figure 3.13(a)	1050	524	5.9Hz
Figure 3.13(b)	747	496	8.1Hz
Figure 3.13(c)	300	302	17.2Hz
Figure 3.13(d)	186	242	30.2Hz

Table 3.1: *Geometrical and computational comparison of the degraded models presented in Figure 3.13.*

[5]Note that the table reports the number of (convex and planar) polygons in a model, whereas Figure 3.13 displays the number of triangles for each model (after undergoing an implicit triangulation process by the Open Inventor package).



(a) **(b)** **(c)** **(d)**

Figure 3.13: *Example reductions for a sample geometric model. The figures show: (a) the original model, (b) a reduced model using a sub-volume one sixty-fourth of the total volume, (c) a reduced model using a sub-volume which is one eighth of the total volume, and (d) a reduced model where the sub-volume equals the entire volume.*

3.4 Summary

As this chapter draws to a conclusion, we now possess tangible solutions to all of the outstanding problems which we identified as essential to the production of a perceptually-based LOD system.

In the first instance, we have developed a means of describing a computer-generated image in terms of its component spatial frequencies (c/deg). The approach which was adopted is based around an image segmentation process in which pixels are grouped into atomic visual features based upon JND determinations (using the CIELUV colour difference formula). These features are then analysed to extract their fundamental relative spatial frequencies (c/pixel); and then finally these values are transformed into units of c/deg based upon the FOV of the display device. During our discussion, we looked at Fourier methods and found these to be inapposite for our purposes: Fourier analysis is a valid technique for qualitative comparisons of image content or frequency filtering operations, but it has severe limitations as an accurate and concise measure for perceived detail.

Following from this, we produced a computational model for contrast sensitivity (Equation 3.21). Building principally from the work of Kelly (1979) and Rovamo and Virsu (1979), we developed a mathematical equation to predict the contrast sensitivity of a standard observer to a stimulus of any size, velocity, and eccentricity. We also illustrated how this model could be used to compute the visual acuity of the observer under the same conditions.

Finally, we formulated a general framework for polygon simplification which addresses the three criteria presented in Chapter 2. That is, the framework provides a mechanism to limit the scale of degeneration for any arbitrary simplification algorithm. This is achieved by segmenting the object's volume into a number of sub-volumes and independently applying a simplification algorithm to all of the polygons entirely within each sub-volume. In order to correctly degrade the surface property information, we apply an area-weighted averaging of the previous surface properties to any new triangulations.

With the above solutions in hand, we can now consider how these may be integrated to produce an efficient computer graphics system which modulates the detail of objects based upon their size, velocity, eccentricity, and perceptual content.