Chapter 4

Implementation

'Jim, we've absolutely got to have those communicators.
Without them we don't have the computers,
And without the computers we don't have a chance!'

(Dr. Leonard 'Bones' McKoy, *Star Trek: Miri*)

We now have sufficient pieces to our puzzle to embark upon a general model and subsequent implementation for a perceptually-based LOD system. We have a suitable measure for the degree of perceived spatial detail in a computer-generated image, and a model to predict the visibility of any part therein. Consequently we now have to consider how these can be combined to produce an efficient implementation which can be usefully integrated into a real-time computer graphics system.

4.1 A General Model for Perceptual LOD

Before proceeding further, it would be beneficial to briefly describe the general model that we are currently constrained to use due to the decisions which we have made thus far.

In the first instance, we are required to compute the spatial frequency content of each object off-line (because we have chosen a more visually accurate image based analysis rather than a geometry based technique). This will involve sampling the object from a number of different viewpoints in order to capture its 3D profile. By performing the spatial frequency analysis off-line we also alleviate much of the computational burden from the on-line scheduler (the code which performs the LOD modulation during the simulation), and hence simplify the time-critical component of our implementation. (For this reason many VE systems have included an off-line stage to perform any possible preprocessing, e.g. Airey *et al.*, 1990; Hitchner and McGreevy, 1993; Maciel and Shirley, 1995; Funkhouser and Séquin, 1993.)

Once on-line, the system must monitor the size on screen, angular velocity, and eccentricity of each object (in units of pixels, deg/s, and deg, respectively). It can then use this information to compute the highest resolvable spatial frequency under those conditions, using our model for contrast sensitivity. Also, given the spatial frequency data that were found from the off-line stage, we can estimate the instantaneous frequency content of each LOD. Finally, the system can then use all of this information to choose the optimal LOD for the object.

It is an obvious point, but one worth noting, that not all objects in a VE will need to be considered by the on-line scheduler. Only *degradable* objects need to be considered: i.e. those objects for which multiple levels of detail exists.

From the above, it is clear that our general model should be split into two fundamental stages: an off-line (preprocessing) stage, and an on-line (scheduling) stage. We can encapsulate the high level structure of each of these stages in the following two algorithms. The remainder of this chapter will be devoted to describing the implementation of these two stages in further detail.

Off-line :

```
produce a number of different LOD models for various objects
...
For each degradable object
   For each LOD of the object
      For a number of sample viewpoints around the LOD
         render the LOD
         extract all relevant spatial frequencies from image
         store perceptual attributes with model definition
      EndFor
   EndFor
EndFor
```

On-line :

```
perform any necessary initialisation
...
At each update of the VE
   For each degradable object
      find the object's size, velocity, and eccentricity
      evaluate the spatial frequency content of each LOD
      calculate the highest visible detail in the object
      choose the optimal LOD to use for the object
   EndFor
EndAt
```

4.2 Rationalising the Model

Thus far we have been concerned with the development of measures and tools which encapsulate contemporary models of visual perception as precisely as possible. However we do not need, or wish, to include this level of complexity in our system. Indeed, given the sheer complexity of our visual system, we could not hope to simulate all of the involved factors accurately and rapidly. Therefore, in order to entertain the notion of a real-time implementation, we are going to have to analyse the data requirements of our application more precisely, and rationalise our model where possible.

4.2.1 Optimising the Off-line Stage

With regards to the spatial frequency extraction process, we are obviously not concerned about every possible spatial frequency that exists within an object. Ideally, we would like to be able to find a single value to characterise the perceptual content of an object for our purposes.

In the first instance, we are not concerned with the absolute range of frequencies in a single object, but only with the *difference* between two successive levels of detail for the object. Take the example of two models of a calculator: one with buttons on the front face and one with no buttons. The spatial frequency of the calculator case is irrelevant because it appears in both models. It is the spatial frequency of the buttons which will dictate when the two models should be switched. Therefore, once we have collected the overall frequencies for all of the LOD models, we can go through and remove instances which are common to successive models, i.e. frequencies that have the same magnitude and location within the image. This will leave us with only the **unique spatial frequencies** for each model.

Performing this data compression can drastically reduce the amount of information that we must process. In addition to this, we can also streamline our model further by undertaking the following minor simplifications:

- **Ignore Contrast** : if we ignore the contrast of a feature (i.e. assume it to be 1.0) then we would effectively modulate detail based upon a user's visual acuity, rather than their contrast sensitivity. This would be a valid simplification, both from a perceptual and a computational standpoint, for the following reasons:

    1. As a result of performing our spatial frequency analysis off-line, we cannot account for any changes in the lighting of the model that may occur during the simulation. By ignoring contrast we therefore conservatively take the worst case scenario; thus helping to preserve our goal of being able to modulate detail without affecting the user's percept of the scene.

       (Also, by ignoring the contrast of a feature we can reduce the disk and memory overheads of our model.)

    2. As can be observed from Figure 3.8(a), our contrast sensitivity drops off precipitously beyond the optimal spatial frequency, e.g. taking the curve for 1 deg/s, if a feature dropped from full contrast (1.0) by an order of magnitude to 0.1, then the threshold spatial frequency would only change by a few units, from $\sim$13 c/deg to $\sim$10 c/deg. We would therefore be losing a relatively small degree of accuracy if we were to favour visual acuity over contrast sensitivity.

- **Ignore Orientation** : the orientation of a spatial frequency is of little importance when considering static targets[1] because the orientation of

[1]Except perhaps to compensate for the aspect ratio of the display.

a feature will not improve its visibility. Orientation only becomes relevant when considering moving targets because the visibility of a moving spatial frequency is dependent upon its alignment with the direction of motion.

For example, consider a long, thin, vertical feature. Its horizontal spatial frequency (derived from its width) will be very high, but its vertical spatial frequency (derived from its height) will be very low. If this feature moves horizontally, then at a certain velocity it will become invisible to the naked eye. If this feature then moves at the same velocity but in a vertical direction, it will remain visible because the vertical spatial frequency is far lower than its horizontal counterpart. This effect can be observed in the natural world: e.g. when you look fixedly out of the window of a fast moving train towards a meshed fence of horizontal and vertical wires, then you cannot see the vertical wires of the fence, only the horizontal wires which are aligned with your direction of motion.

Therefore, incorporating orientation into our model would only prove beneficial under circumstances where certain, significantly elongated features are travelling at high velocity and are perpendicularly aligned to the direction of movement. Even if this case is met, there may be other features in the object which do not meet these criteria and hence are visible, thus the level of detail of the object as a whole cannot be altered. We therefore propose that the computational effort required to account for feature orientation and direction would not be justified. Accordingly, we will ignore the orientation of spatial frequencies in our implementation. That is, we take the worst case scenario for each feature in an image, i.e. the lowest frequency that exists at any orientation in a feature.

4.2.2 Optimising the On-line Stage

Most of the data reduction that can be performed on our model is relevant to the off-line stage. As a result of this optimisation, we automatically optimise the on-line stage because the complexity of the data it must handle is reduced. There is therefore comparatively little rationalisation required for the implementation of the on-line stage. However, there is one algorithmic simplification that we will regard as reasonable to impose:

- **Ignore Rotational Velocity** : we have already noted that there are two distinct types of velocity: translational velocity and rotational velocity (see Section 3.2.4, page 94); where rotational velocity is the rate at which an object rotates about its axes. Modulating spatial detail using rotational velocity would be an involved process and one which does not fit easily into an object-based selection paradigm such as LOD. Also we have the problem that very little perceptual data have been reported to help us correctly account for the visibility of a rotating object.

  For example, when an object is rotating, then the translational component of any feature's velocity will vary for different phases of the rotation (e.g. the translational velocity component of a point on a rotating sphere varies sinusoidally as the sphere turns). Also, features further away from the point or axis of rotation will travel faster over the display (e.g. if the radius of the rotating sphere was increased, then the point would have to travel over a larger area in the same time period). We can therefore see that different features in an object could be travelling at different velocities across the retina. We could endeavour to find the worst case scenario once again and simply use this to model the effects of rotational velocity. However the worst case would be represented by the translational velocity of a feature located at the point/axis of rotation, which will be 0 deg/s.

  To be accurate, we would require a system that could manage detail at a feature or polygon level, rather than an object level. Such a system would require the tracking of each individual feature within each level of detail of each object; projecting the relevant 3D coordinate for each feature into screen space and calculating velocity based upon the feature's previous such value. It is extremely likely that such a technique would be computationally prohibitive.

  For all of these reasons we feel that a general model for seamlessly modulating an object's spatial detail based upon its rotational velocity cannot be achieved using the standard notion of LOD. We will therefore not attempt to incorporate this into our current implementation.

4.3 Implementation of the Off-line Stage

We have now produced a general model for perceptually modulated LOD, and rationalised this model for a real-time computer graphics system. We can now consider how this model may be implemented. Accordingly, we will here discuss the various issues that are associated with the implementation of our off-line stage. That is, the generation of a range of LODs for different objects as well as the sampling and processing of each LOD's (relative) spatial frequency profile. The descriptions in this section should provide the reader with sufficient insight into the exact operation of the off-line stage, and how it can be implemented.

4.3.1 Generating each Object's LOD

Naturally, in order to modulate the detail of an object, we need to be able to create different representations of that object at various complexities. Section 2.2 provided a review of numerous polygon simplification techniques which can be used to produce these levels of detail from an original object. Any of these could be used for the task of generating different LODs for objects. However, this thesis makes the recommendation that some framework should be enforced to restrict the spatial extent of a reduction. One such scheme was developed in Section 3.3.

The reason for this recommendation is that we need to be able to create a collection of objects that contain different ranges of unique spatial frequencies. If all levels of detail for an object contain unique spatial frequencies of the same magnitude, then there will be no situation when one model can be favoured over another based upon their perceptual content. Essentially, we require a collection of LOD models with a range of *visual* complexities, in addition to a range of *geometric* complexities.

4.3.2 Generating each LOD's Spatial Frequency Profile

The major component of the off-line stage (as far as this thesis is concerned) is the computation of each object's perceptual content: that is, the relevant spatial frequencies which are component in each LOD model. These data are then attached to the object's description (along with its other attributes such as geometry, materials, position, etc.) for later use by the scheduler to select the best LOD in any situation.

4.3.2.1 Gathering the Spatial Frequency Data

Before we can investigate the spatial frequency information within the rendered image of a model we must think about how we are going to present the model. There are a number of display considerations that we must address:

1. **Model Size** : how large should we display the object on the screen? If we display an object too small then we may lose some of the small detail differences between levels of detail. The obvious answer is to render the object as large as possible on the display device.

   More precisely, we wish to position the object at a distance from the viewpoint so that the object's **smallest bounding sphere**[2] projects to the largest possible screen area, with no part of the sphere projecting to a point out with the viewport. This placement will ensure that all of the object will be displayed on the viewport no matter what its orientation.

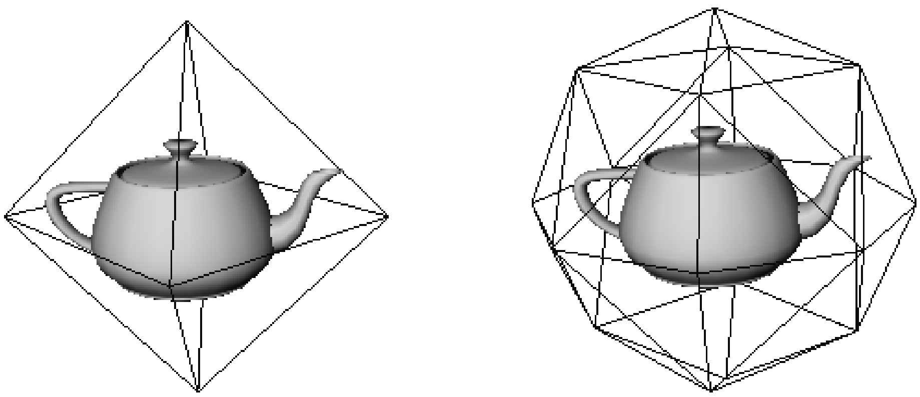
   A related issue here is also how to specify the size of an object (because we concluded in Section 2.1.7, Page 38, that size is a more resilient measure than distance for modulating detail). This is an arbitrary decision because the value will only be used in a relative sense, and so an absolute measure is not required. We will therefore favour the definition of object size used by Roehl (1995), because this offers a lightweight and orientation invariant measure. That is, the size of an object is represented by the length (in pixels) of the projected radius of an object's smallest bounding sphere. We record the size of an object when it is sampled because this information will be required by the on-line stage.

2. **Sample Viewpoints** : we have already noted that because our spatial frequency analysis is based upon a single 2D snapshot of a model at one particular orientation then we must endeavour to take multiple snapshots around the model in order to capture its 3D profile. But which viewpoints

[2] We define a 'bounding sphere' for our purposes as any sphere which completely encloses an object, and whose centre is coincident with the centre of that object. We will augment this term with the adjective 'smallest' when we wish to imply the singular such sphere which encompasses an object the most tightly.

should we choose to sample the model from, and how many viewpoints should we use?

Given the optimal distance between the object and the viewpoint that we have just resolved, the set of all possible viewpoints around that object will lie on the bounding sphere whose radius is equal to this distance (the so called **view sphere**). However, there is an infinite number of points on a sphere, and thus an infinite number of potential viewpoints. We therefore attempt to discretise the space of possible views around an object by sampling a subset of points on its bounding sphere. This can be achieved by using an appropriate **spherical tessellation** routine to produce a uniform and consistent distribution of viewpoints (e.g. Borgefors, 1992; Hwang and Yang, 1993). Figure 4.1 illustrates this concept by showing two possible samplings for an example object.



(a) **(b)**

Figure 4.1: *Two examples of the space of possible sample viewpoints around an object. In each case, the viewpoint locations are denoted by the vertices of the tessellated bounding sphere. All viewpoints are assumed to be oriented towards the centre of the sphere. (a) illustrates the simplest case of 6 sample viewpoints, while (b) shows a more complex example with 18 viewpoints.*

It is interesting to note that Maciel and Shirley (1995) used a similar method in their LOD system. They sampled the bounding hemisphere of an object off-line, calculating images at each sample point in order to choose the best model to display at any orientation. (Presumedly, Maciel and Shirley used a bounding hemisphere—rather than a complete sphere—because they were concerned only with objects that were positioned on a ground plane, and so the underside of these objects would never be visible.)

The question that we are left with is therefore: how many viewpoints should we sample around an object? This is difficult to specify optimally beforehand because the number will depend upon the visual complexity of the object in question (which it is the purpose of our analysis to eventually ascertain). Also, we will be constrained to only certain numbers of sample viewpoints, dependent upon the particular tessellation scheme which is used to triangulate the bounding sphere.

The simplest such tessellation would produce 6 sampled viewpoints, giving samples for the top, bottom, and four sides of the object, as in Figure 4.1(a). This is a crude sampling, but may be sufficient for some simple objects such as cubes, cones, spheres, etc. However, for more complex objects a higher sampling resolution will be required. In general, between 18 and 128 sample viewpoints provide the most effective results.

Choosing a larger number of samples will not necessary affect the performance of the system during a simulation; however it will substantially increase the time taken to generate the object's spatial frequency profile, and it will also increase the storage overheads for the object.

3. **Lighting Conditions** : the degree of lighting which is used can affect the visibility of features in an object. For example, taking the extreme case where there is no lighting at all, then no features will be visible in an object. We should therefore attempt to sample the object under lighting conditions similar to those that will be experienced during the simulation, or under the brightest lighting conditions if these will vary on-line.

   It is important to note that this need not be done to an accurate degree. Because we ignore the contrast of features, it is only necessary that a feature be visible; even if only slightly. Therefore, we find that as long as there is ample ambient lighting then this should be sufficient for our purposes.

4.3.2.2 Processing the Spatial Frequency Data

Having collected the spatial frequency information (c/pixel) for a number of 2D snapshots of the object around its view sphere, we then filter these data to produce a more manageable description. This will involve applying certain operations to each individual snapshots in isolation, and also applying operations based upon comparative judgements between snapshots of more than one LOD from the same viewpoint. We shall refer to these as **local filtering** and **global filtering** operations respectively. Taking the first of these, we can describe the processes involved in the local filtering phase as:

Local Filtering Operations :

1. **Contrast Filtering** : we have decided to ignore the effects of contrast; effectively assuming a contrast of 1.0 for all features. This simply means that we disregard all contrast values that were found by the spatial frequency analysis (or more efficiently, we instruct the analysis program to not spend time computing the contrast values in the first place).

2. **Orientation Filtering** : we have also decided to ignore the effects of orientation in our model. We should therefore record only the lowest spatial frequency for all sampled orientations in each feature. This means that we use the longest part of a feature to represent its size and so we never underestimate the feature's perceived size.

The result of applying these local filtering operations is to produce a spatial frequency profile which contains a single value for every feature in each 2D snapshot. This value provides a measure for the threshold frequency of that feature which is both orientation invariant and contrast invariant. However, we can improve upon this data compression and reduce the spatial frequency profile to a single value for an entire 2D snapshot by applying the following global filtering operations.

Global Filtering Operations :

1. **Locate Unique Spatial Frequencies** : we are only concerned with the spatial frequencies that differ between successive levels of detail. We can therefore remove all frequencies that exist with the same magnitude and position in successive LODs at each viewpoint[3].

   Once this has been done, we can then discard the position information from the remaining spatial frequencies as we no longer require these data.

[3]The efficacy of this process is aided if we use a polygon simplification technique that incorporates a mechanism to limit the spatial extent of a degeneration. The reason being that all polygons above a certain size will never be modified and so the resulting spatial frequencies that they contribute towards are normally unaffected between successive models.

2. **Locate the Threshold Spatial Frequency** : our ultimate goal is to find the single value for spatial frequency which defines the point at which the next LOD model should be chosen. We can specify this as follows: given two successive LOD models, l and $l+1$ (where $l+1$ is more detailed than l), we wish to find the lowest unique spatial frequency in model $l+1$ which is greater than the highest unique frequency in model l (if one exists).

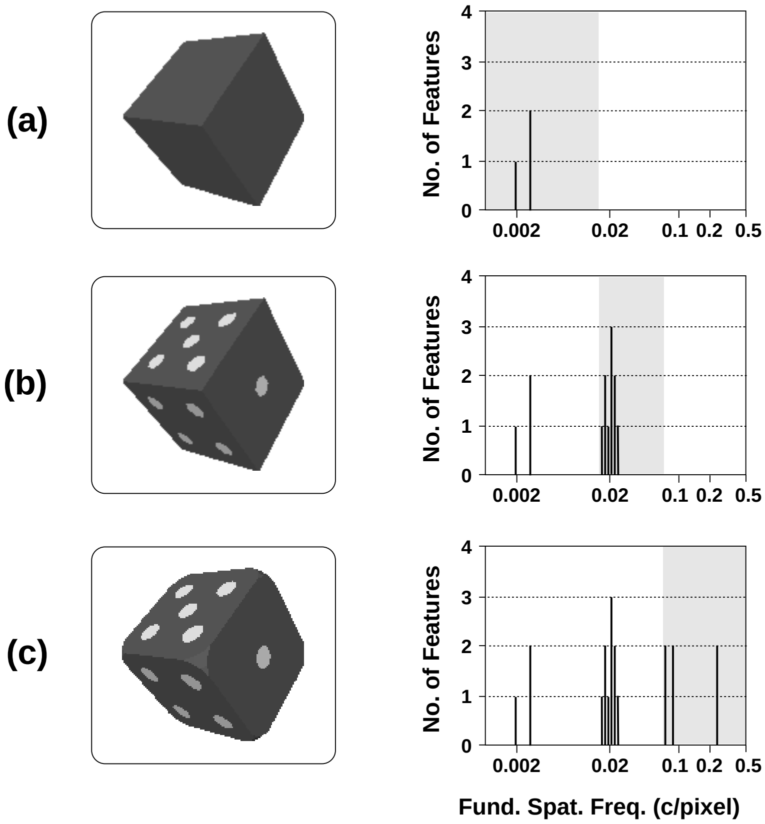


Figure 4.2: *An example illustration of the transition spatial frequencies for the die example we used previously in Figure 3.7. The shaded regions on the graphs show the frequencies over which the particular model can be used without visual defect. The value of spatial frequency which marks the boundary between two shaded regions represents the transition frequency for those two models.*

We shall refer to this value as the **transition spatial frequency** (α_{trans}) between models l and $l+1$. When we decide that the user is able to perceive spatial frequencies of this magnitude, then we must switch from the lower detail model (l) to the higher detail model ($l+1$). (N.B. if we have n levels of detail for an object, then we will always have $n-1$ transition spatial frequencies for each viewpoint.)

In simple terms, this procedure tries to keep the object at the lowest possible LOD. A more complex model is only chosen when the detailed features that it adds become potentially visible to the user.

Figure 4.2 illustrates the result of all these filtering operations on the die example we used in the previous chapter. We can see here that the transition spatial frequencies are determined by the lowest unique frequencies between two successive LODs, e.g. the value of α_{trans} between the lowest LOD, Figure 4.2(a), and the middle LOD, Figure 4.2(b), is around 0.015 c/pixel.

4.3.3 Focussing the Off-line Stage

Figure 4.3 attempts to encapsulate all of the various phases that contribute towards the off-line stage of our model. The overall result is to find a single value of spatial frequency (c/pixel)—for each sampled viewpoint—which defines the point when the next level of detail should be selected. Each of the five phases illustrated in Figure 4.3 are described below:

(a) **LOD Generation** : two different levels of detail are produced from the original die model. The models are produced so that the first model (left) only has very small features removed (the rounded edges), whilst the second model (right) also has larger features removed (the spots).

(b) **Sample Viewpoint Rendering** : both LODs are rendered from a number of viewpoints around the object's view sphere. We also record the projected size of the object at this stage so that we know the object size to which all the spatial frequency data pertain. The example here shows only 3 viewpoints for reasons of clarity. Note that these 3 viewpoints must correspond for all LODs, e.g. the first snapshot of the left-hand model was generated using the same viewpoint as the first snapshot of the right-hand model.

(c) **Spatial Frequency Analysis** : each 2D snapshot image is analysed to assess its spatial frequency content using the image segmentation algorithm that we developed in Section 3.1. Once this has completed, we will have segmented each image into a number of visual features, and for each feature we will know its position (x,y), spatial frequency (α), contrast (c), and orientation (θ).

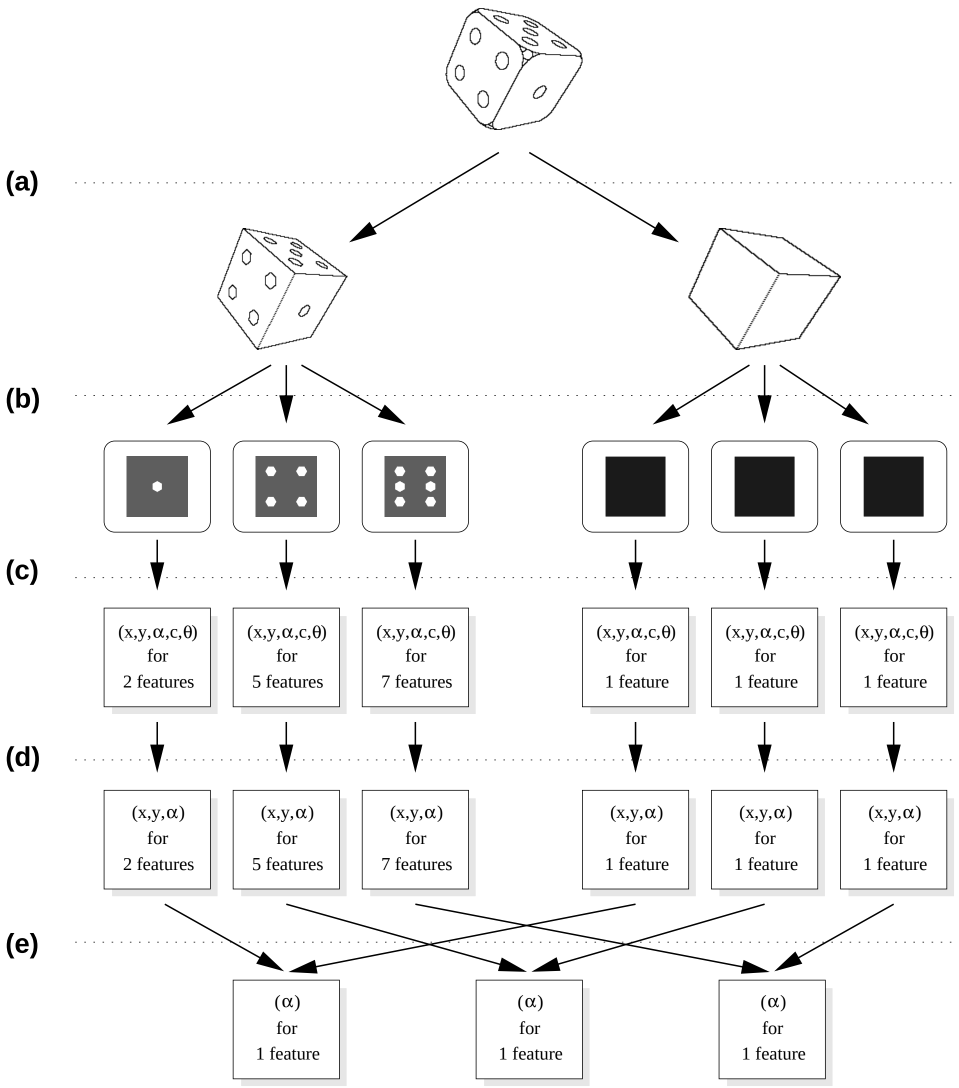


Figure 4.3: *An example overview of the off-line stage, showing the generation and spatial frequency analysis of two levels of detail of an original model. Phases (a)–(e) are described in the main text.*

(d) **Local Filtering of Data** : for each spatial frequency profile we remove the contrast data and find the lowest orientation value for each feature. This leaves us with only the position and spatial frequency attributes for each feature.

(e) **Global Filtering of Data** : we compare the data for both levels of detail at equal viewpoints. In each case we remove all frequencies which are common to both models and then discard the position information. Finally we find the threshold spatial frequency which defines when we should switch from one model to the other. This produces a single value of spatial frequency for each sampled viewpoint.

4.4 Implementation of the On-line Stage

Having discussed the details of the off-line stage, this next section will deal with the various implementation issues which are associated with the on-line stage. This will principally involve describing the task of selecting the optimal LOD for each degradable object, i.e. how to assess the perceptual content of each model on-line, how to evaluate the visibility of the object, and finally how to use this information to select the most appropriate LOD for an object.

4.4.1 Initialisation

The initialisation phase simply involves loading the results from the off-line stage and accounting for the specific viewing conditions, i.e.

- Load all of the spatial frequency information for each viewpoint of each LOD. (These values are currently in units of c/pixel.)
- Calculate the field of view of the display. This will generally be known for a head-mounted display, and can be calculated for a desktop display using Equations 3.2 and 3.3 (Page 77).
- Scale all of the relative spatial frequency values into units of c/deg based upon the specific FOV. This procedure is described in Section 3.1.6.

4.4.2 Monitoring Object Attributes

Our model of level of detail allows us to select an object's representation based upon its size on the display device, its angular velocity, and the degree to which it exists in the peripheral field. Our first task must therefore be to calculate these attributes of an object.

Object Size (σ pixels) : we have already stated on Page 112 that we will use the length of the projected radius of an object's smallest bounding sphere as a measure of object size. This can be calculated by finding the projected (screen) coordinate of the object's centre, and the projected coordinate of a suitable point on the object's smallest bounding sphere. We then simply calculate the Euclidean distance between these two points.

It should be noted that not just any point on the bounding sphere can be used to calculate the radius. We must choose a point which projects onto the circumference of the sphere's 2D projection, i.e. a point such that the 3D radius is perpendicular to the 3D vector connecting the viewpoint and the centre of the object.

Object Velocity (v deg/s) : we are concerned with the velocity of an object across the user's retina. We therefore need to find the rate at which the object moves across the display device (pixels/s), and then scale this into units of deg/s using the FOV information that we calculated during the initialisation phase.

We already know the current position of the object on the display device (we found this from our size calculation above). If we therefore keep a record of the previous screen position and a timestamp for that position, then we can calculate the velocity (pixels/s) of the object as:

$$\frac{|currOrigin - prevOrigin|}{currTime - prevTime}, \tag{4.1}$$

where $currOrigin$ and $prevOrigin$ are 2D coordinates supplied in units of pixels, and $currTime$ and $prevTime$ are given in units of seconds. This value can then be scaled into units of deg/s in the same way that the c/pixel values for spatial frequency were scaled into c/deg.

Object Eccentricity (E deg) : finally, we need a measure for the degree to which the object exists in the user's peripheral field. This is simply the angular distance between the object and the **focus point** (e.g. the point on the display which the user is gazing towards). Once again we can find this value in units of pixels and then scale this into units of degrees based upon the display FOV.

We must be careful when calculating this value: that is, we must not forget the fact that an object is not normally a point source. For example, if we measure the object's eccentricity by simply taking the distance between the focus point and the projected centre of the object, then there will be parts of the object which are closer to the focus point. As a result we run the risk of underestimating the visibility of these parts of the object.

Our measure of eccentricity must therefore be based upon the distance between the focus point and the projected point within the object which is closest to the focus point. We can approximate the latter by choosing the point on the object's smallest bounding sphere which is closest to the focus point. To improve efficiency, we could choose this value for our size calculation and hence reuse that value here (see Figure 4.4 below).

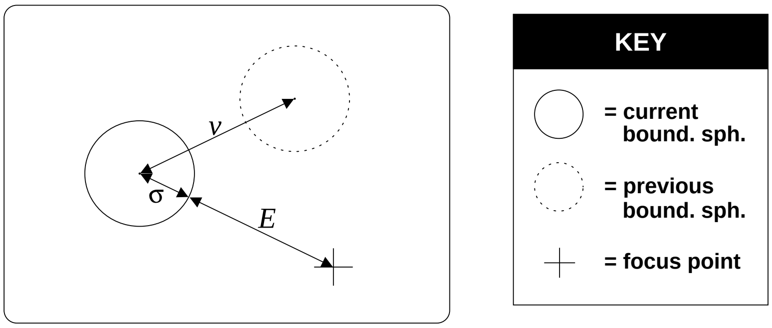


Figure 4.4: *Illustrating the pixel lengths used for the computation of object size (σ), velocity (v), and eccentricity (E). The two spheres depict the projected smallest bounding sphere of the object for the current and previous frames.*

4.4.3 Evaluating Object Frequency Content

Now that we know the size (σ), velocity (v), and eccentricity (E) of an object, we can continue with the next phase of our model. The goal of this phase is to assess the perceptual content of an object's various LOD models. This will involve using the results from the off-line stage in order to approximate the transition spatial frequencies (α_{trans}) for the object based upon its current orientation and size.

4.4.3.1 Finding Transition Spatial Frequencies for Arbitrary Orientations

We know the precise transition spatial frequencies of the object for a number of sampled viewpoints around the object (i.e. for a number of sampled orientations of the object). We wish to use this information in order to estimate the transition frequencies of the object for any arbitrary orientation.

Firstly we must resolve the orientation of the object relative to the viewpoint. This simply means finding the difference between the orientation of the 3D vector connecting the object and the viewpoint, and the orientation of the object in 3D space. This tells us the orientation of the object as perceived by the user.

We then estimate the transition spatial frequencies of the object at that orientation by interpolating between the closest sampled viewpoints around the view sphere (e.g. Renka, 1984). For example, if we used a triangular tessellation of the sphere to generate the viewpoints, then we could find the three closest sampled viewpoints and interpolate between these to find the correct value of transition spatial frequency.

Alternatively, a more efficient (but less accurate) method would be to use a point sampling technique where we find the sampled orientation which is closest to the object's current orientation and simply take this value as the result for that orientation. Of course the accuracy of this method will improve as we increase the number of viewpoints that are sampled.

4.4.3.2 Factoring Object Size into the Transition Spatial Frequencies

The values for transition spatial frequency that we have managed to compute so far are only valid for an object which is the same size as when it was sampled during the off-line stage. However, we can easily generalise our model for objects of any size by scaling our result with the relative size of the object.

Consider the example of a book sitting on a desk. If this book was then moved away until its size on your retina was exactly half the size of the original, then all of the features within that object will also be half their original size, e.g. the writing on the cover of the book. By corollary, all of the spatial frequencies within the book will have doubled (because spatial frequency is inversely proportional to size). We can therefore see that in order to incorporate object size into our model we should multiply the present transition frequency values by the fraction,

$$\frac{\sigma_0}{\sigma}, \tag{4.2}$$

where σ is the current size of the object on the display device (pixels) and σ_0 is the size of the object when it was sampled during the off-line stage (pixels).

We now know the instantaneous values for transition spatial frequency (α_{trans}) within an object. These values tell us the threshold frequencies at which we can select the different LOD models for the current view of the object. All we require now is the ability to ascertain the degree of spatial frequency which the user can perceive in the object under its current viewing conditions.

4.4.4 Evaluating Object Visibility

The goal of this phase is to estimate the **highest visible spatial frequency** (α_{vis}) that we expect the user to be able to perceive in an object. This is done by applying our model of contrast sensitivity using the values of object eccentricity (E) and velocity (v) that we have already calculated.

4.4.4.1 Applying our Visual Acuity Model

Earlier, in Section 3.2.3 (Page 93), we described visual acuity in terms of our contrast sensitivity model using the relation:

$$H(\alpha, v, E) - 1 = 0, \tag{4.3}$$

where the function $H(\alpha, v, E)$ is defined in Equation 3.21. We wish to use this relation by supplying values for v and E and then subsequently solving the equation for α, thus giving the highest spatial frequency which is visible under those spatiotemporal conditions.

Unfortunately we have already noted that an analytical approach to solving this relation for α would be computationally involved, and that an iterative method would be more elegant (see Section 3.2.3, page 93). We therefore advocate an interval halving algorithm here. This proceeds by instantiating the v and E parameters with the computed values for object velocity and eccentricity, respectively. Then an initial value for α is taken and this estimate repeatedly refined until the resulting value of $H(\alpha, v, E) - 1$ is within a desired tolerance of zero. A sufficiently large initial value for α should be chosen so that the highest root of $H(\alpha, v, E) - 1 = 0$ is always found, e.g. the highest spatial frequency that our visual system can resolve, α = 60 c/deg (Campbell and Gubisch, 1966).

Naturally such an iterative calculation should be avoided during a real-time simulation for performance reasons. So we propose that values from this process should be precomputed when the simulation is first started and the results stored in a look-up table (LUT) for optimum performance.

4.4.4.2 Defining the Highest Displayable Spatial Frequency

The above model would be sufficient for evaluating the visibility of objects in the real world; however we must take into consideration the fact that the user is viewing a computer display, and so the angular resolution of that display will limit the size of detail which they can be exposed to. We can incorporate this factor into our model by introducing the notion of a **Highest Displayable Spatial Frequency**, ξ. This characterises the highest frequency which can be displayed by an output device, and will of course depend upon the field of view and pixel resolution of the device.

Referring to Section 3.1.5.1 (Page 76), we recall that a single pixel can be considered to be half a contrast cycle. Therefore the number of contrast cycles that can be displayed on a device is equal to half its pixel resolution. This can be transformed into a value of spatial frequency once we know the FOV of the display using the following simple equation:

$$\xi = pixelRes/(2 \times angularRes), \tag{4.4}$$

where $pixelRes$ represents the number of pixels in the display and $angularRes$ represents the angular field of view of the display (in degrees).

As part of the rationalisation of our model, we resolved to ignore the orientation of spatial frequencies in our implementation. We can therefore describe our final definition of highest displayable spatial frequency as the largest value between the horizontal and vertical resolutions of the display device, i.e.

$$\xi = \max\left(\frac{horizPixels}{2 \times horizFOV}, \frac{vertPixels}{2 \times vertFOV}\right), \tag{4.5}$$

where $horizPixels$ and $vertPixels$ are the horizontal and vertical pixel resolutions of the display, and $horizFOV$ and $vertFOV$ are the horizontal and vertical angular resolutions of the display. (Refer to Equations 3.2 and 3.3, Page 77, on how to calculate values for $horizFOV$ and $vertFOV$ for a desktop display.)

4.4.4.3 Completing our Visual Acuity Model

Now that we can define the highest displayable spatial frequency for a display device, we can incorporate this factor into our model. We do this by simply thresholding our present value for highest visible frequency (calculated from our contrast sensitivity model) with the appropriate value of ξ (calculated from the display resolution), i.e. we take the smallest of these two values (see Figure 4.5). This means that our model will now tell us the highest visible spatial frequency that can be perceived at a particular velocity and eccentricity, taking into consideration the viewing conditions and display device limitations. We shall refer to this value as α_{vis}.

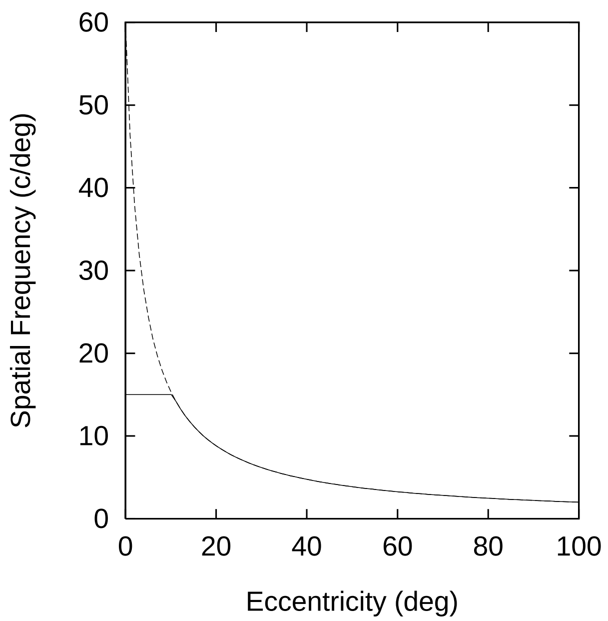


Figure 4.5: *The effect of incorporating maximum displayable spatial frequency into our model. The curves show the decline of spatial frequency sensitivity with eccentricity only (using Equation 3.20 with M_0 = 60 c/deg). The broken line illustrates the original model, whilst the overlayed solid line indicates the effect of thresholding the model with ξ = 15 c/deg.*

It may be instructive to note that our term, highest displayable spatial frequency, is merely a simplification of the display device's **modulation transfer function** (MTF). The MTF is a measure of a display's ability to maintain the contrast of a signal as a function of its spatial frequency (Evans, 1990), i.e. in effect it is the electronic equivalent of our biological system's contrast sensitivity function (CSF). The general operation would therefore be to threshold the CSF using the MTF. However, because we are only dealing with visual acuity here—i.e. the uppermost point of the CSF—we only need use the equivalent uppermost point of the MTF. It is this point on the MTF which we have called highest displayable spatial frequency (ξ).

4.4.5 Selecting the Optimal LOD

As a result of the preceding calculations, we now know the highest spatial frequency that the user should be able to perceive in an object (α_{vis}) and the effective transition frequency (α_{trans}) between each level of detail of that object. It is now a simple matter to find the best LOD to use for that object, e.g. the lowest LOD such that the α_{trans} value for the immediately higher LOD is greater than the limit of vision for the object, α_{vis}. That is, the least complex model such that the smallest visual change that it will entail will be below the user's threshold of vision.

Here we make the assertion that the value of α_{trans} associated with model l is for the transition between models $l-1$ and l, where model $l-1$ is defined to be less complex than l. Of course, if all models' α_{trans} value are below α_{vis}, then the highest LOD should be selected (because this means that selecting any model other than the highest LOD will produce a visible switch).

The following section of C++ source code illustrates the implementation of the LOD selection routine, given the values for α_{trans} and α_{vis}. These two values are referred to in the code segment as the array `object.SFtrans[]` and the scalar `object.SFvis` respectively.

```
int LOD = object.highestLOD;
while ( LOD > object.lowestLOD && object.SFtrans[LOD] > object.SFvis )
   LOD--;
object.setLOD( LOD );
```

It is worth noting one significant optimisation which can be made here, now that we have covered all the appropriate material. This follows from the observation that we may not need to calculate the transition spatial frequency between every LOD. For example, if we eventually select the highest LOD then we only need to know the transition frequency associated with that model. In the above code segment we can therefore replace the array look up on the previously computed α_{trans} value (`object.SFtrans[LOD]`) with a class method that actually performs the transition frequency calculation *in situ* and returns the resulting α_{trans} value. This way, if the algorithm selects the highest LOD and so exits the while loop on the first iteration, then we only compute one transition frequency.

4.4.6 Focussing the On-line Stage

Figure 4.6 illustrates how the various phases of the on-line scheduler can be integrated into the standard **Sense–Process–Display** loop of a typical VR system. The position of the scheduler within the Process stage is important in order to gain the optimal performance. It must follow any processing which updates the location and velocity of each object (including the viewpoint); but it should precede any subsequent processing (e.g. collision detection, dynamics equations, etc.) so that this will be applied to the actual model about to be displayed.

To collate all the material that we have just discussed with regards to the on-line stage, we will now take all of the phases indicated by the right flow diagram in Figure 4.6 and outline the major operations involved in each of these.

(a) **Find Object's Size, Velocity, and Eccentricity** : the first action we must perform is to calculate the size of an object in screen space (pixels), its angular velocity based upon its previous position (deg/s), and the eccentricity of the point on the object's smallest bounding sphere which is closest to the focus point (deg). In the case of the latter two attributes, we first calculate these in terms of pixels and then transform these into values in terms of degrees based upon the display FOV and resolution.

(b) **Evaluate Transition Frequencies** : this phase attempts to encapsulate the spatial frequencies within each of the object's levels of detail. We have reduced this representation to a single viewpoint-dependent and

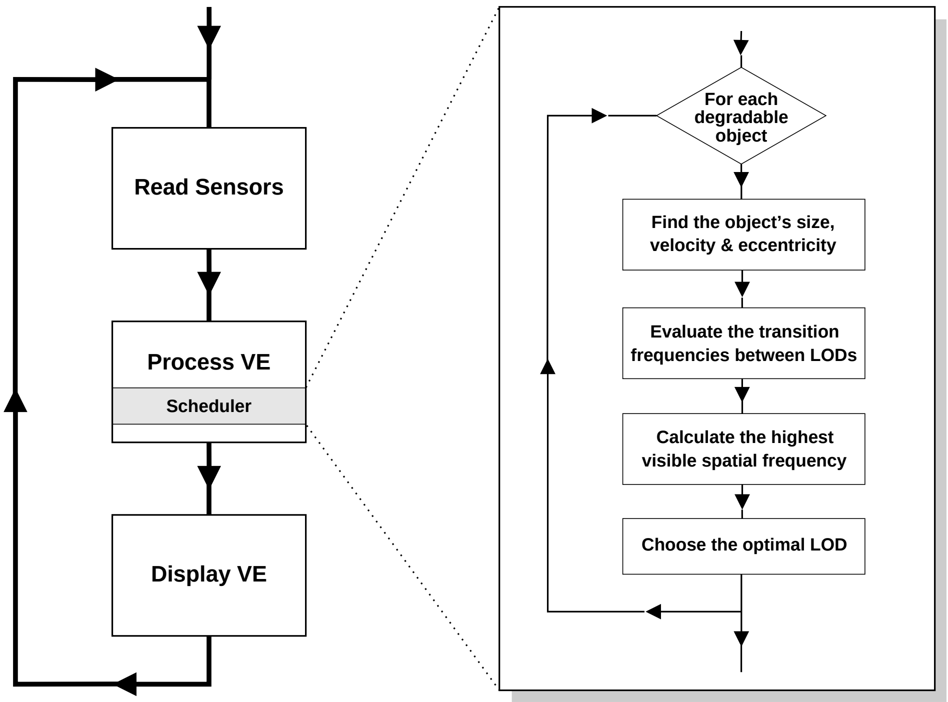


Figure 4.6: *Illustrating how our LOD scheduler (right-hand flow diagram) fits into the main loop of a typical VR system (left-hand flow diagram).*

size-dependent value (α_{trans}) between each LOD. This value defines the threshold spatial frequency that describes the size of the smallest feature which differs between successive LODs. We calculate this value for the current view of the object by using the sampled values from the off-line stage to estimate each LOD's α_{trans} for the perceived orientation, and then scale these results based upon the relative size of the object on the display device.

(c) **Calculate Highest Visible Frequency** : this process attempts to estimate the degree of spatial detail which the user can perceive in the object, based upon its current velocity and eccentricity. That is, the highest spatial frequency (α_{vis}) that we would expect them to be able to resolve in the object. This value is calculated from our model of visual acuity (which in turn is derived from our contrast sensitivity model), and the spatial frequency limit of the display device.

(d) **Choose Optimal LOD** : finally, we can use the value of α_{vis} and the values for α_{trans} between each LOD to select the optimal LOD for that object such that the user should not be able to perceive any visual change. This is the lowest LOD where all visual changes between this model and the highest LOD are below the user's threshold of vision.

It should be clear how our model manages the optimal LOD of an object based upon its velocity and eccentricity, however it may be useful to clarify why our model also finds the optimal LOD to display based upon the object's size on the display device. To do this, we will take the example of a large blackboard with a piece of chalk resting on its ledge. Let us say that the chalk is displayed such that it is 10 c/deg long and that the highest displayable spatial frequency of the output device is 15 c/deg. If the blackboard is moved away so that it becomes half the original size, then the chalk will be have twice the spatial frequency, i.e. 20 c/deg. Ordinarily this may still be visible to the human eye, but because the display's maximum limit is 15 c/deg, we know that the chalk is too small to be displayed. Effectively, by incorporating the notion of a highest displayable spatial frequency (ξ) into our contrast sensitivity model we gain the ability to calculate when a feature projects to less than a single pixel.

4.5 Summary

This chapter has introduced a general model for perceptual LOD, and subsequently described its efficient implementation. We have seen that our model must be split into two principal stages: an off-line and an on-line stage. The off-line stage is where the various levels of detail for an object are generated, and where these are analysed to assess their spatial frequency content. The on-line stage is composed of a scheduler that selects the optimal LOD to use at each update of the simulation. This is done through a process of estimating the actual spatial frequency of the object from the off-line data, and predicting the visual acuity of the user with regard to the object.

We proposed several simplifications to our model in order to produce a feasible real-time implementation. However, none of these simplifications should be taken as inexorable constraints. It would be possible to increase the complexity of the implementation (e.g. incorporate and act upon orientation information) if more compute time were available, or if the real-time constraint were lifted (e.g. for a passive animation sequence).

We will defer discussion of the general issues surrounding perceptual LOD until a later chapter. However, the particular approach that we have adopted raises a number of issues which are worth identifying here. The following points therefore provide a discussion for the most prominent of these issues.

- **Orthogonality** : each of the three components of our model can be used individually or in any desired combination. For example, we can disable the modulation of detail with respect to eccentricity by simply instantiating all eccentricity values to 0 deg. Our model will then still function correctly, but will only modulate detail based upon the size and velocity of an object. Similarly we can disable velocity optimisations by instantiating all velocity values to 0 deg/s, and disable size optimisations by defining the relative size of an object (Equation 4.2) as 1.0.

- **Eye Tracker Support** : we have already commented that our visual perception is based upon the attributes of stimuli projected onto our retinae, and that to be completely accurate we should therefore incorporate an appropriate eye tracking technology into our system.

  Our model is sufficiently general to encompass this facility, but not require it. That is, the notion of a focus point can be used to represent the position of a user's gaze on screen, or it can simply be defined as a fixed location if eye tracking is not available (e.g. the centre of the screen).

- **Resolution Adaptability** : we sample each object at a specific resolution to find its spatial frequency content. This therefore raises the question of what happens if the simulation is run at a different resolution to that of the off-line sampling stage?

  In the first case, if different resolutions are chosen for the on-line and off-line stages, but these have the same aspect ratio, then this will cause no problems. For example, we could sample objects at a resolution of 640×480 pixels and then use a simulation resolution of 1280×960 or 320×240. This is possible because simply changing the resolution of the display does not alter the degree of detail which is displayed within an object at a certain size: it simply means that we can display objects at a larger (or smaller) screen size.

  If the aspect ratio between the on-line and off-line resolutions differ then we could simply scale all transition spatial frequencies appropriately or, more accurately, we could repeat the spatial frequency sampling process for that on-line resolution.

On a related issue, if we change the FOV of the display but retain the same pixel resolution then this will also cause no problems. This is because our model automatically compensates for any FOV by delaying the conversion from c/pixel values into c/deg until the simulation is actually begun.

- **Limitations of the Model** : flexibility and accuracy have been prime objectives during the development of our model. However, there comes a point when these two factors must be traded off against each other. The biggest limitation which results from the particular approach that we have adopted is that we cannot compensate for any environmental changes during a simulation (e.g. changes in lighting, fog, time-of-day, etc.). This is because our assessment of an object's perceptual content is based upon data which are collected off-line.

  As we have already noted, this is not a severe limitation because we can simply ensure that we collect an object's spatial frequency profile under ideal conditions (e.g. high illumination, no fog, etc.). This ensures that our model will function accurately under all environmental conditions, although it may not function as optimally under substantially degraded conditions.

Up to this point our discussion has been mostly conceptual in content. As a result we are left pondering a number of questions regarding the implementation of our model. For example, to what extent can our model reduce lag in a VR system? How accurate is our visual acuity model? Does our model succeed in its goal of modulating LOD with no perceptual artifacts? These and other issues will be addressed in the next chapter which presents an evaluation of our implementation for perceptually modulated level of detail.