Computer Science
Technical Report

Colorado State University

A 3D Visualisation System for the Image Understanding Environment

Karthik Balasubramaniam

June 19, 1998

Technical Report CS-98-108

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

A 3D Visualisation System for the Image Understanding Environment

Karthik Balasubramaniam

June 19, 1998

Abstract

An interactive 3D visualisation system 3DCAMS (3D Camera Modeller and Simulator) has been developed within the DARPA Image Understanding Environment (IUE). The system supports interactive visualisation of 3D models represented as IUE spatial objects. It provides the user a range of choices for camera modelling including abstract modelling, as well as physical camera simulation. The system supports camera models of different levels of complexity. The perspective pinhole camera model, Tsai's camera model, and physical lens models are supported. It is intended that the system provide support for 3D tasks in the IUE such as camera calibration and camera model simulation using ray tracing. The system also serves as a testbed for the tractability of the IUE C++ software development environment in developing a reasonably large interactive application.

1 Introduction

The "Holy Grail" of computer graphics has long been the synthesis of photo-realistic images of the 3D world. The goal of computer vision, on the other hand, is to understand the 3D world by making 3D inferences from 2D images. In both fields, much revolves around the camera model in use.

In computer graphics as well as in computer vision, one needs to know certain properties of the camera imaging process. This knowledge may range from the basic; such as magnification, focal length, focussed distance, to the complex; such as radial distortion coefficients, aperture and lens parameters, sampling uncertainty, etc. There is the notion of an *abstract camera model*, which is a simplification of the physical camera, reduced to a convenient mathematical model. Examples of abstract models include the pinhole model [FD82], thin-lens and thick-lens approximations [JW57], and photogrammetric models [STH80]. These models usually lead to convenient closed form equations representing the imaging process. Then, there is the notion of the *physical camera model*, which typically involves physical simulation based on principles from optics and radiometry. This approach yields models that are more faithful to the imaging process in the real world at the expense of greatly increased complexity. This thesis focuses on physical simulation of camera geometry and compares results for different camera models, abstract as well as physical. The various camera models mentioned above are discussed in Section 2.

Section 3 on physical camera modelling describes the implementation of a ray tracer that simulates image formation by physical lenses in a camera's lens system. The thin lens model is a simplified model of a physical lens which ignores its thickness. When this is not justifiable, as is the case for many common camera lenses, the thick lens model is used. The treatment covers thin lens and thick lens ray tracing, which simulates the imaging process for such lenses.

Abstract camera models may be obtained through the process of *camera calibration*. When a real camera is used in computer vision applications, one needs to obtain the camera's imaging model in order to extract inferences from the acquired imagery. The process of calibration determines a best-fit abstract model given a set of 3D-2D point correspondences. Section 4 on camera calibration focuses on a widely used calibration technique; Tsai's algorithm [Tsa87].

A significant component of this thesis work is a 3D visualisation system, which allows a user to interact with different types of camera models. A suite of camera modelling, camera manipulation and rendering routines are implemented in an interactive 3D visualisation system, 3DCAMS (3D Camera Modeller and Simulator). The design and use of the GUI-based system is described in Section 5. The 3DCAMS visualisation system supports traditional 3D model visualisation via OpenGL rendering, and also offers rendering with physical camera models (using ray tracing). 3DCAMS also offers an interactive camera calibration component based on Tsai's calibration technique.

Section 6 describes the software architecture of the 3DCAMS system. 3DCAMS has been developed within the DARPA Image Understanding Environment (IUE) [Mea92, BDHR91]. The IUE is a C++ software development environment for computer vision and imaging research. 3DCAMS uses the IUE class libraries for representing and manipulating its spatial objects, and for 3D transformation operations. This section describes the object-oriented architecture of the 3DCAMS system, and evaluates the role of the IUE.

2 Camera Modelling

This section discusses different approaches to the problem of reducing a camera's imaging process to a mathematical model. The abstract models using pinhole, thin lens, and thick lens approximations are introduced. Concepts in lens optics are introduced, and image formation is described for each of the models. This section provides the background from optics necessary to understand the use of these image formation models in synthetic image generation by ray tracing, which the following section addresses.

2.1 The Pinhole Perspective Camera Model

A commonly used camera model is the pinhole perspective projection camera model which is an abstract model used to represent perspective projection. The model is illustrated in Figure 1. A point in object space is imaged onto a point on the camera's image plane. The image point is the intersection of the ray joining the object point and the *centre of projection*, with the image plane. The "pinhole" appellation arises out of the centre of projection being an infinitesimal point, which pinholes in box cameras try to emulate. All object points are perfectly imaged as single points on the image plane.

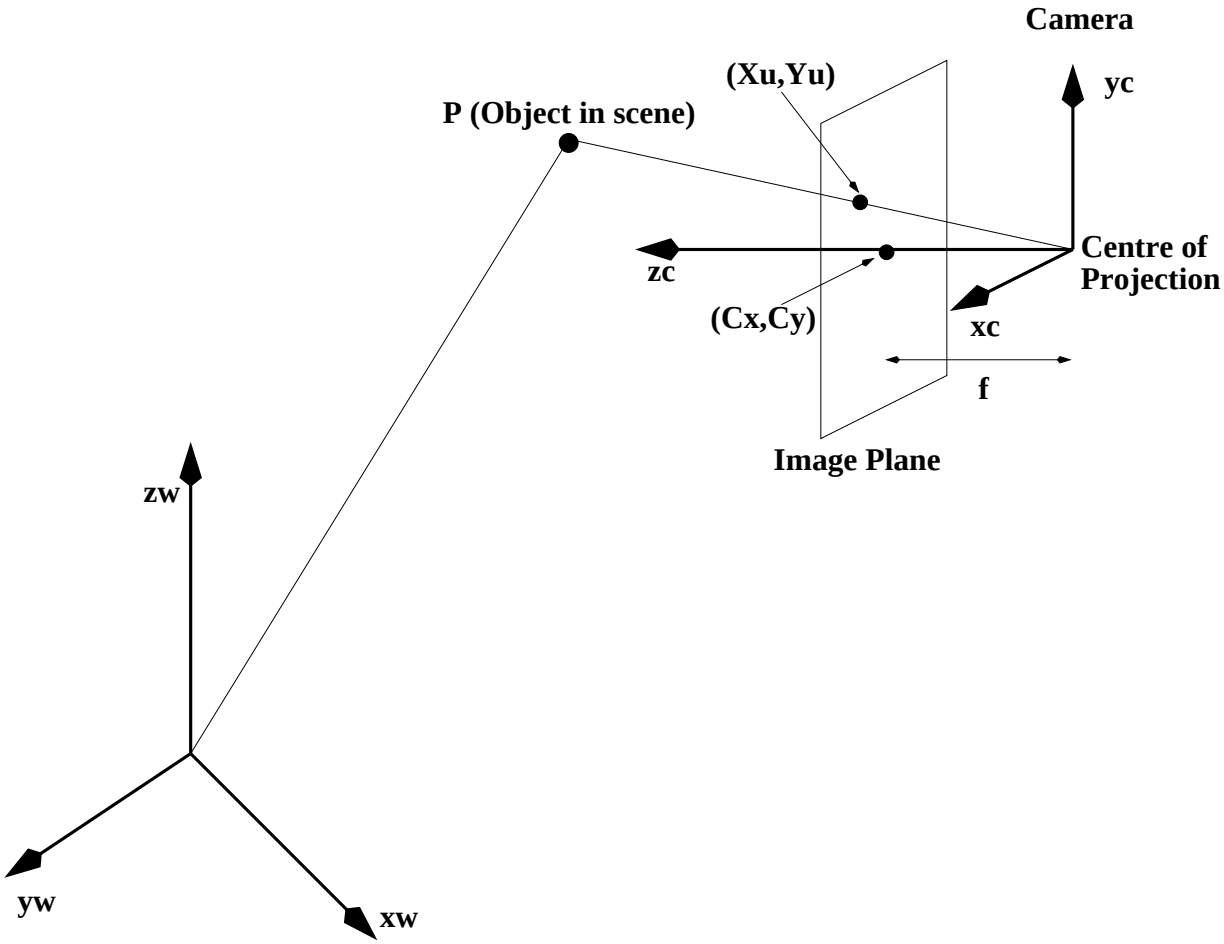


Figure 1: The perspective projection camera model

The *focal length* is the distance from the camera's centre of projection to the image centre. We have

$$X_u = f\frac{x_c}{z_c} \tag{1}$$

$$Y_u = f\frac{y_c}{z_c} \tag{2}$$

where X_u and Y_u are 2D coordinates of the object point projected onto the image plane; x_c, y_c, and z_c are 3D coordinates of the object point in the camera coordinate system.

The pinhole model is typically used in graphics perspective projection pipelines. In more exacting photogrammetric applications, *lens distortions* not modelled by the pinhole representation (there is no concept of a lens in this representation) are often significant, and hence must be taken into account. An example of such distortions is *radial distortion*, which has the effect of translating image points radially towards the centre. This effect is modelled in a more elaborate version of the pinhole model with additional parameters, known as Tsai's camera model, to be described in Section 4.

The 3DCAMS visualisation system, which contains implementations of the camera models described in this thesis, supports the pinhole camera model. Polygonal objects are displayed in either wireframe or shaded form using OpenGL. Polygons may be shaded using either flat or Gouraud shading, possibly combined with texture mapping.

While the pinhole camera is indeed simple, it does not address many phenomena that significantly affect the way real world cameras acquire imagery. Real cameras need to use lenses to focus light rays onto the film and cannot achieve the perfect point-to-point imaging of the pinhole camera. In real world cameras, there is the problem of portions of the imaged world being "out of focus". This is an artifact that is inherent in real lenses, and must be modelled if photorealism is desired. It is also often necessary to use more complex camera models with more parameters. For instance, one may need to accommodate aperture effects, adjustable perspective (as in zoom lenses), accurate radiometry or other effects. As shall be seen, these issues are relevant in accurately modelling the imaging process. Bearing these issues in mind, the discussion now shifts to the role of lenses in image formation.

2.2 Thin Lens Model

A thin lens may be defined [JW57] as one whose thickness is small in comparison with the distances generally associated with its optical properties. Such distances are, for example, the *primary and secondary focal lengths* and object and image distances. Figure 2 illustrates the notion of focal points. Every thin lens in air has two focal points, one on each side of the lens and equidistant from the centre. *Paraxial rays* (rays parallel to the optical axis) approaching from object space converge at the *secondary focal point* F' after refraction through the lens. Similarly, paraxial rays from image space converge at the *primary focal point* F. The distance between the centre of a lens and either of its focal points is its focal length.

Image formation is illustrated in Figure 3. The position of the image of an object is given by [JW57]

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'} \tag{3}$$

where s is the object distance, s' is the image distance, and f is the focal length, all measured from the centre of the lens. Note that symbols referring to object space entities (in front of the lens) are unprimed, while those referring to image space (behind the lens) entities are primed. Any pair of object and image points such as P and P' are called *conjugate points*. Planes through these points perpendicular to the optical axis are called *conjugate planes*.

A simple numerical example will help illustrate the thin lens image formation model. Let an object point be located 6 cm in front of a convex lens of focal length 4 cm. From 3 we may solve for s':

$$s' = \frac{sf}{s - f} \tag{4}$$

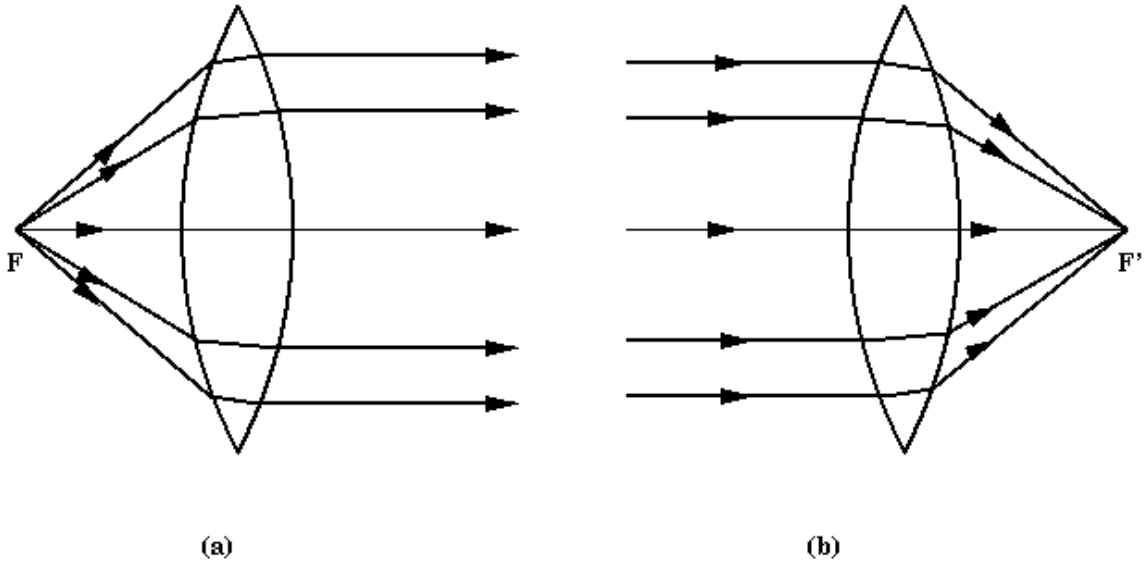


Figure 2: Ray diagrams showing the primary and secondary focal points of a thin lens

Direct substitution of the given quantities in this equation yields

$$s' = \frac{(6) \times (4)}{(6) - (4)} \tag{5}$$

or s' =12.0cm. The image is formed 12cm behind the lens, and is inverted, as shown in Figure 3. The figure illustrates the geometric method of locating the image of the object point. Use is made of the fact that a paraxial ray is refracted by the lens towards the focal point, and that a ray passing through the lens centre passes through undeviated. When two such rays originate from the object point, their intersection behind the lens defines the image point. The focal points are thus useful in geometrically modelling image formation by the lens. The vigilant reader may have noticed that the geometry so far has been confined to a single cross-sectional plane. It is easy to see that by symmetry, this plane may be rotated about the optical axis without affecting the arguments in any way. Thus, object points are 3D points, and their image points are located in 3-space as well. This is not to be confused with 2D image points as in image points on an image plane. The concept of an image plane with respect to lens image formation is introduced in the following section which covers lens simulation. Here, reference has been made only to linear distances in the cross-sectional 2-space of the figures in order to keep the geometry simple.

2.3 Thick Lens Model

When the thickness of a lens cannot be considered as small compared to its focal length, the thin-lens formulae are no longer applicable. The lens must be treated as a *thick lens*. Thick lens approximations and equations may be found in standard optics textbooks [BW64, JW57, Lai91]. I summarise the salient concepts.

The behaviour of a thick lens is modelled by its focal points and *principal planes*, which are illustrated in Figure 4. Paraxial rays approaching from object space converge at the *secondary focal point* F'' after refraction through the lens. Similarly, paraxial rays from image space converge at the *primary focal point* F.

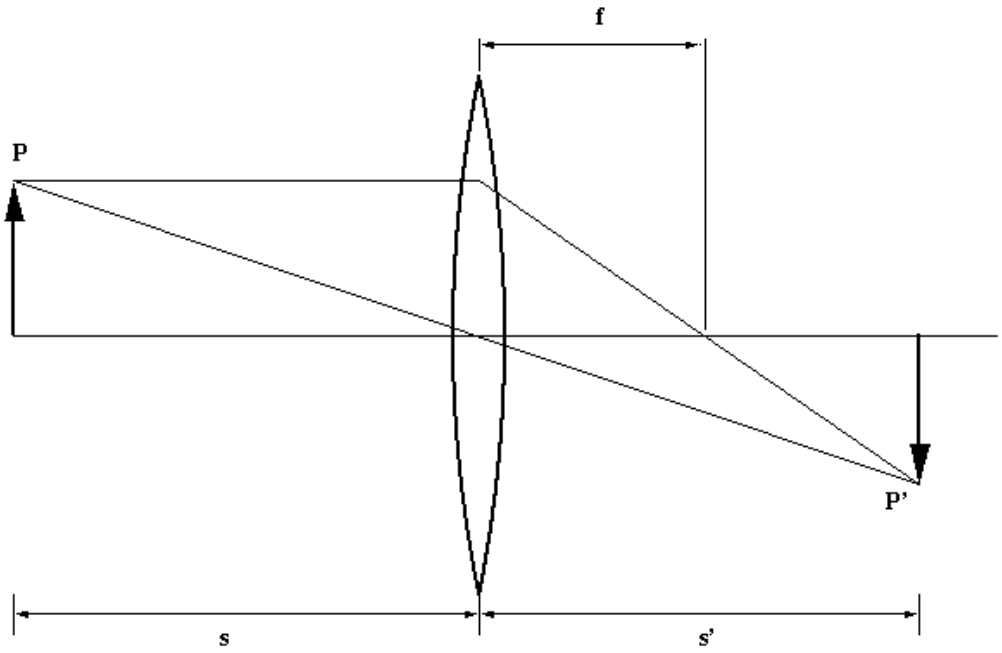


Figure 3: Thin lens image formation model

In either case, the intersection of incident and emergent rays defines a principal plane. There are thus two principal planes, primary and secondary, at which refraction is assumed to occur. It may be shown [JW57] that

$$\frac{1}{s} + \frac{1}{s''} = \frac{1}{f} = \frac{1}{f''} \tag{6}$$

Image formation is illustrated in Figure 5. The distances s and s'' are measured from the principal points H and H'' respectively. Note that symbols refer to entities in object space, lens space, or image space, and are accordingly unprimed, primed or doubly primed. The distance from H to H'' is the *effective thickness* of the lens. Note that when the thickness is taken to be zero, *i.e.* the principal planes are taken to coincide, one is presented with the previously encountered thin lens equation. A geometrical understanding of image formation by a thick lens is aided by an examination of Figure 5. Image formation is similar to the previously described thin lens image formation, except that here all rays in the region between the principal planes are drawn parallel to the optical axis. This follows from the fact that the principal planes have *unit positive lateral magnification* with respect to each other. In other words, each is an upright identical image of the other. There are formulae [JW57] which may be used to compute the location of the principal planes, and the focal points. The image location may then be determined as illustrated in 5. It should be borne in mind that the treatment of lens models so far aims to impart an understanding of the principles involved in image formation by lenses. Computer simulation of image formation through ray tracing, as will be discussed in Section 3, uses a rather different approach.

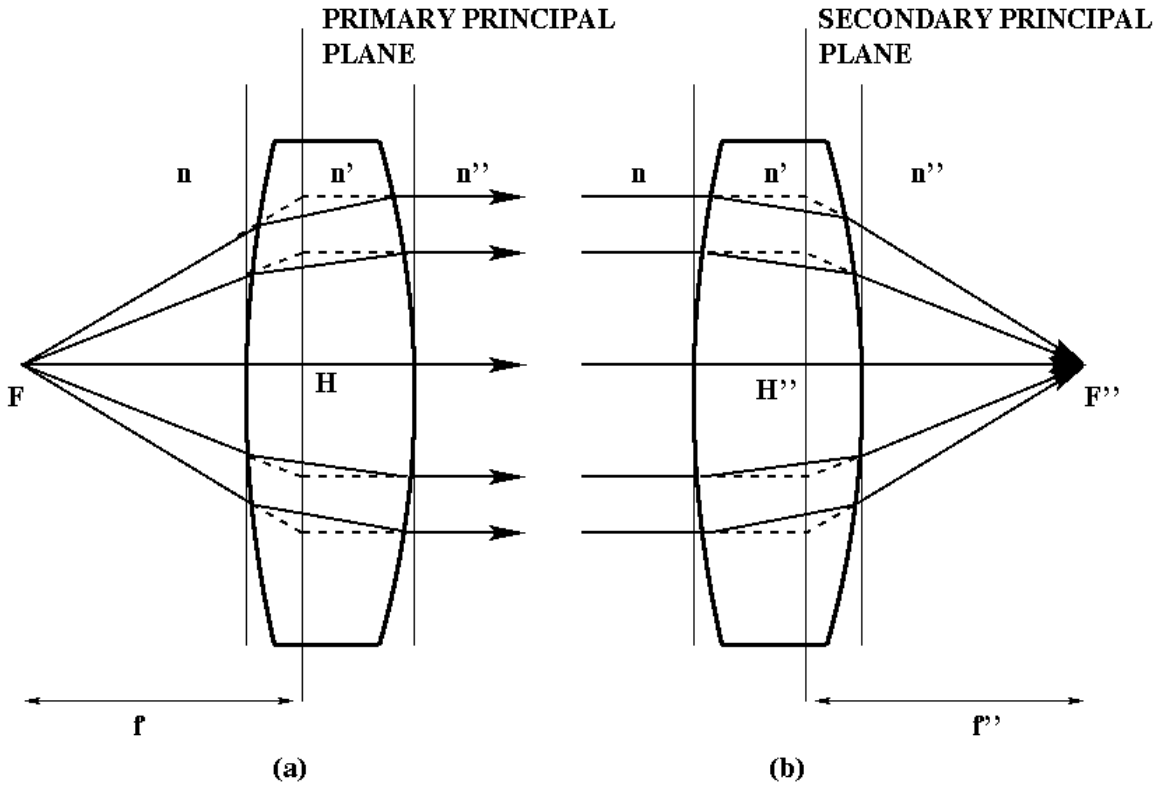


Figure 4: Ray diagrams showing the primary and secondary principal planes of a thick lens

2.4 Physically Accurate Camera Modelling

Camera models in current visualisation systems are geometrically approximate. In general they are not suitable for approximating the behaviour of a physical camera and lens system, although they might be capable of achieving seemingly photorealistic effects. In the following section the focus shifts from abstract modelling to physical modelling. A physical simulation of a lens system may be achieved by tracing ray paths through the lenses [KMH95], applying some of the principles discussed in the earlier sections. The number of lenses in most lens systems usually being small, this should prove to be comparable in expense to regular ray tracing. This approach and its implementation in 3DCAMS, is discussed at length in the following section on physical camera simulation.

In applications where it is required to precisely register captured imagery with synthetic imagery, the camera model must accurately model the actual physical camera. It may be desired to test whether the extracted model of the world matches what is supposed to be observed. Using a physically accurate camera model, it should be feasible (in constrained environments, such as small indoor scenes) to generate accurate synthesised imagery that is closer to captured imagery than would be possible using a simplistic camera model.

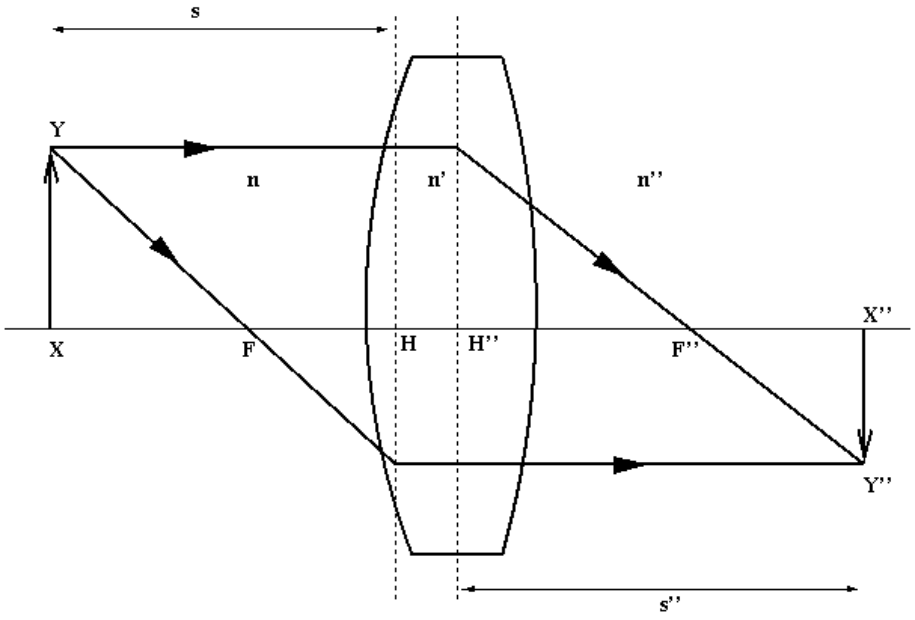


Figure 5: Image formation by a thick lens

3 Simulating a Physical Camera

The preceding section dealt with the mathematical or abstract representation of the camera's imaging process using pinhole, thin lens, and thick lens approximations (in increasing order of photorealism). I use the term photorealism to refer to the extent to which the computer graphic simulation is faithful to the actual imaging process. Alternatively, photorealism is often used to signify *perceived* realism, in which case photorealistic rendering is taken to refer to the synthesis of realistic "effects". This section describes the application of these models in simulating a physical camera using ray tracing. The brute force approach of simulating a camera lens system by treating individual lens elements as transparent raytraceable objects, is also described. A sample scene is rendered by each method and the generated images are compared to illustrate the differences between them.

3.1 Pinhole Ray Tracing

The traditional ray tracing technique [App68, Gla89, Wil94b] uses a pinhole model as shown in Figure 6. The basic idea is to simulate the illumination-reflection process in the 3D world. Being faithful to this process by tracing light rays from the source is impractical as very few rays out of an infinite population actually reach the observer. Instead, light rays are traced from the observer through the pixel. In order to compute the final colour of the pixel, the interactions of the ray with objects and light sources are determined in reverse chronological order. Ray-world interactions include reflection, refraction in the case of transparent objects, shadowing etc.

3.2 Depth of Field

The commonly used pinhole camera model images every object point to a perfect point on the image plane. However, real world cameras (the human eye, for instance) have a finite lens aperture. Each point in object space is imaged as a circle on the image plane. This circle is called the *circle of confusion* (Figure 7). The

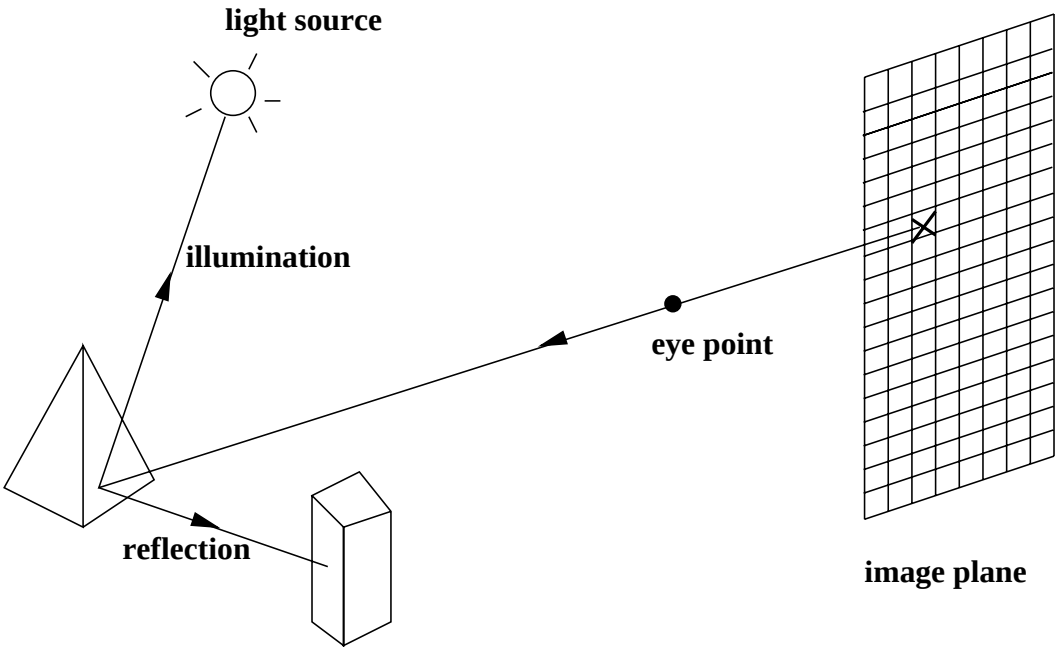


Figure 6: Pinhole ray tracing

size of this circle depends on the object point distance and lens properties. Thus, objects, depending on their distance from the lens, appear either in or out of focus. This effect is known as *depth of field*. Depth of field can be an unwanted artifact in real-world photography, but it is an effect that must be taken into consideration for accurate camera modelling and image synthesis.

3.3 A Post-Processing Approach

Potmesil [PC81] simulated depth of field using a postprocessing technique. The 3D scene is first rendered with a pinhole camera model. All objects are thus in sharp focus. Later, each imaged object point is convolved with a filter whose size is the circle of confusion. Such a postprocessing approach, while capable of generating seemingly realistic effects, can never be truly correct. This is because visibility is computed only from the centre of the lens. The view of 3D object space is different from different points on the lens surface, and thus precludes a correct treatment of the hidden surface problem by a postprocessing approach.

3.4 Thin-Lens Ray Tracing

A better solution to the depth of field problem was presented by Cook [CPC84]. Depth of field occurs because the lens is a finite size, with a nonzero aperture. For any point on the image plane, the computation of final perceived colour needs to integrate contributions from different points on the lens. The view of the 3D world, as seen from a point on the lens, is that obtained by looking at a certain point on the *focal plane* (Figure 7). This point is the conjugate of the image point, as defined in the thin-lens image formation model (Section 2.2). The focal plane is located at a distance s_f, given by

$$s_f = \frac{s'_f f}{s'_f - f} \tag{7}$$

from the thin-lens equation

$$\frac{1}{f} = \frac{1}{s} + \frac{1}{s'} \tag{8}$$

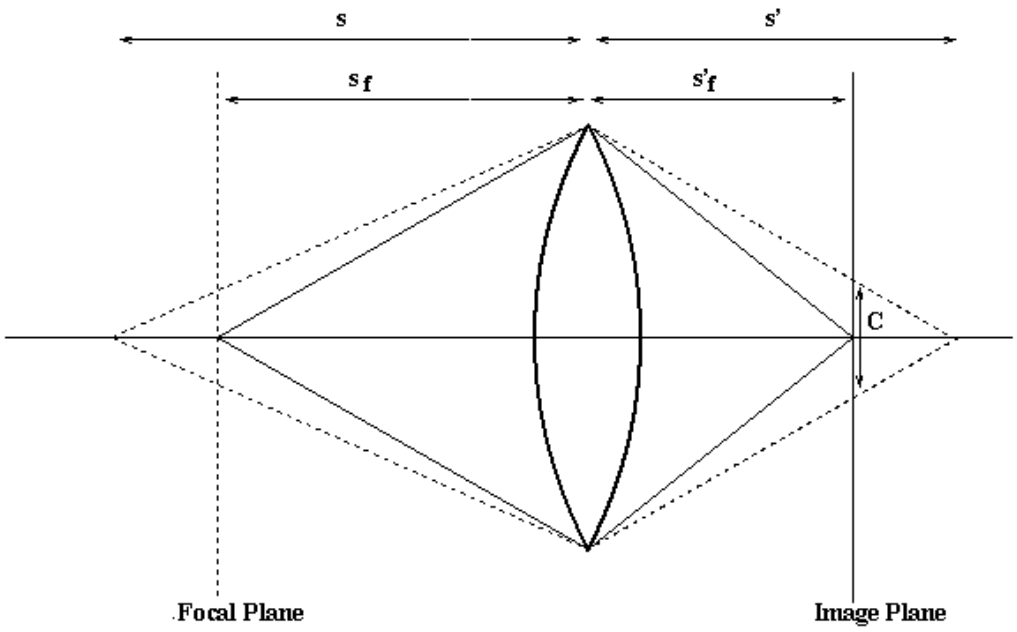


Figure 7: Depth of field and the circle of confusion

s being measured in object space, and s' in image space.

From 8, one may see that points, on a plane that is a distance s from the lens, will focus at an image plane distance s', given by

$$s' = \frac{sf}{s - f} \tag{9}$$

The diameter of the circle of confusion is given by [PC81]

$$C = \frac{|s' - s'_f|d}{s'} \tag{10}$$

where d is the diameter of the lens. An examination of this equation reveals something that one's geometric intuition readily confirms. As the image plane is moved away from the position given by s'_f, the circle of confusion grows larger. Conversely, as an object point moves further away from the focal plane, its circle of confusion grows larger. In other words, its image grows more "out of focus".

Depth of field can be calculated by starting with the traditional ray from the lens centre through the point p on the focal plane. For the additional rays, a sample point on the surface of the lens is chosen and the ray from that point to p is traced. This method integrates depth of field with visible surface determination, and hence gives a more accurate solution to the depth of field problem than Potmesil's approach. Cook[CPC84, Coo86] originally named this technique *distributed ray tracing*. Distributed computing having come into increasing vogue, the technique is now referred to as *distribution ray tracing*.

3.5 Lens Systems

The pinhole and thin lens ray tracing techniques described above are not very useful when it comes to simulating an actual physical lens system of the kind encountered in real cameras. The thickness of lenses usually cannot be considered negligible, thus disfavouring the use of the thin lens model. The alternatives are to use a thick approximation for the configuration of lenses comprising the camera lens system, and the

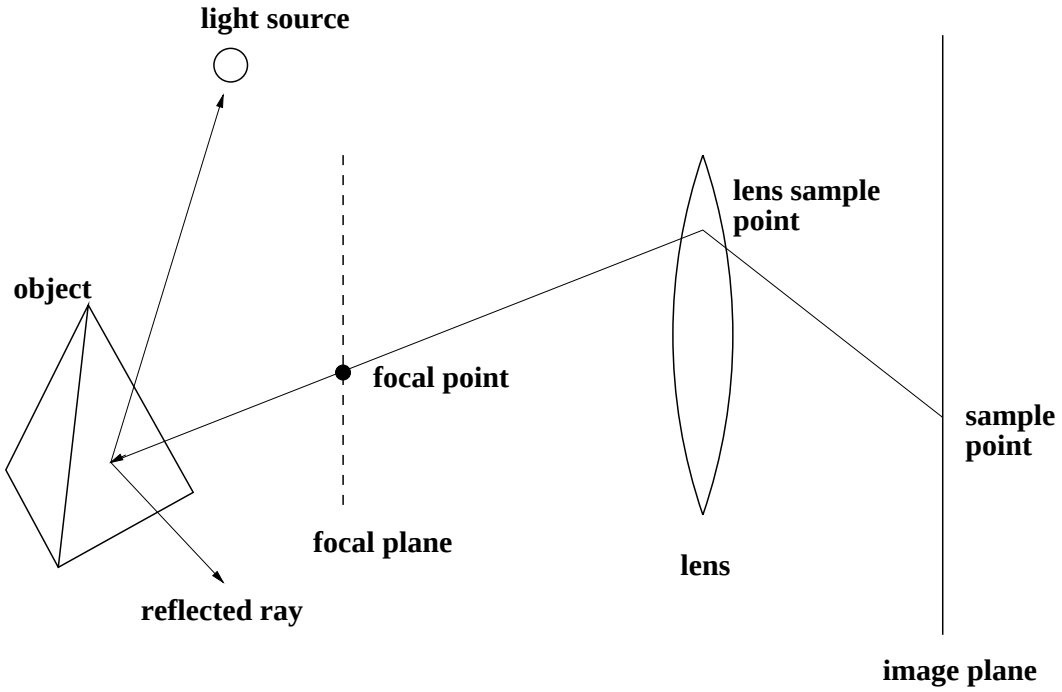


Figure 8: Distribution ray tracing using the thin lens model

more thorough albeit expensive approach of actually ray tracing the individual lens surfaces. These two approaches are dealt with in the following sections. We now look at a typical camera lens system and its specification.

Camera lens systems usually consist of individual spherical glass lenses and *stops* centred on a common optical axis. A stop is an opaque element with a circular aperture used to restrict the passage of light. The specification of a lens system is traditionally presented in a tabular format, such as in Table 1. Each row specifies a lens element surface. The first column is the signed radius of curvature of the spherical lens surface. The radius is omitted in case of a planar surface. A positive radius of curvature indicates that the lens is convex when viewed from object space. Similarly, a negative radius of curvature indicates that the lens is concave when viewed from object space. The next column is thickness, which gives the distance from the current surface to the next surface. Note that in the sixth row of Table 1, no radius is specified. The entry refers to an adjustable diaphragm, the thickness subject to change. In such cases, the specification is a snapshot of the system at a particular point in time. The column titled $R.I.$ denotes the refractive index of the material on the side of the surface facing image space. The last entry is the diameter of each lens element, or its aperture. The lens system in Table 1 is an example of a *double-Gauss* lens [KMH95, Smi92], this version being used in $35mm$ cameras. This lens system is shown in Figure 9.

3.6 Ray Tracing a Lens System

A robust and accurate method of simulating image formation by lenses is to trace light rays through the system [KMH95]. The propagation of a ray through a lens system involves both finding the point of intersection between the ray and the surface and the refraction of the ray as it crosses the interface between the two media (usually glass and air). This computation is usually simple, since most lenses are either spherical or planar. Since the visibility ordering of the lens surfaces is known *a priori*, no search is required to determine the closest intersection. The expense of tracing the extra rays is small compared to the total cost of object intersection tests and illumination computations. The basic procedure is outlined below.

Radius(mm)	Thickness(mm)	R.I.	Aperture(mm)
58.950	7.520	1.670	50.4
169.660	0.240		50.4
38.550	8.050	1.670	46.0
81.540	6.550	1.699	46.0
25.500	11.410		36.0
	9.000		34.2
-28.990	2.360	1.603	34.0
81.540	12.130	1.658	40.0
-40.770	0.380		40.0
874.130	6.440	1.717	40.0
-79.460	72.28		40.0

Table 1: Parameters for an example lens system

```
R = Ray(point on image-plane → point on rear-most lens element)
for each lens element E_i, rear to front:
    p = intersection(R, E_i)
    if(p is outside aperture of E_i)
        R is blocked, discard R
    else
        test for refraction
        compute new direction of R
```

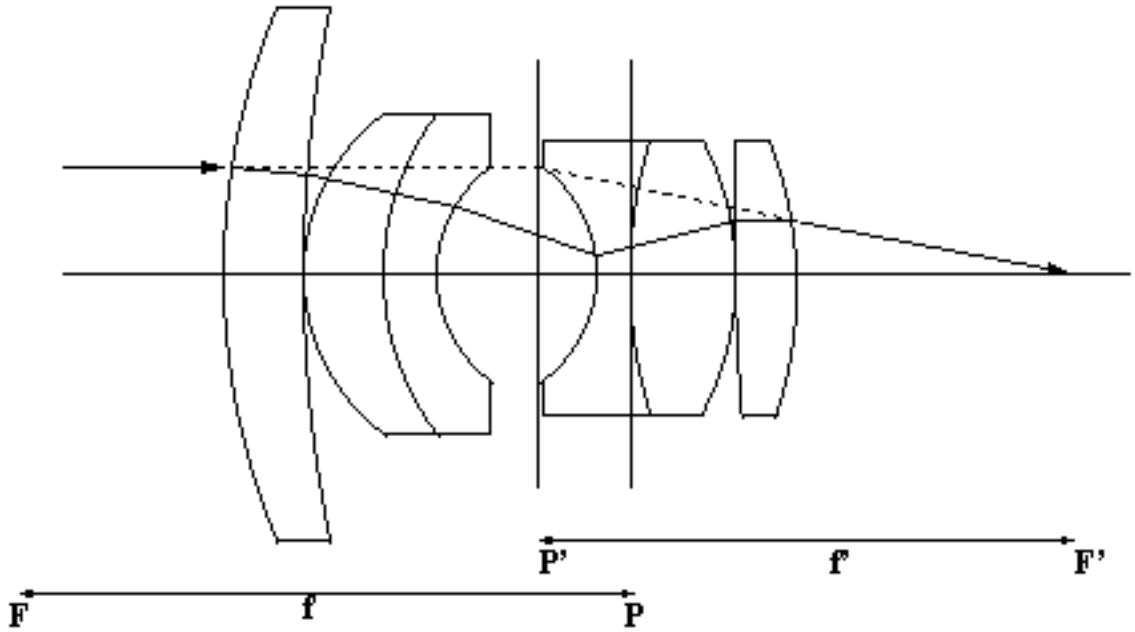


Figure 9: Ray tracing a lens system

3.7 Using a Thick Lens Approximation

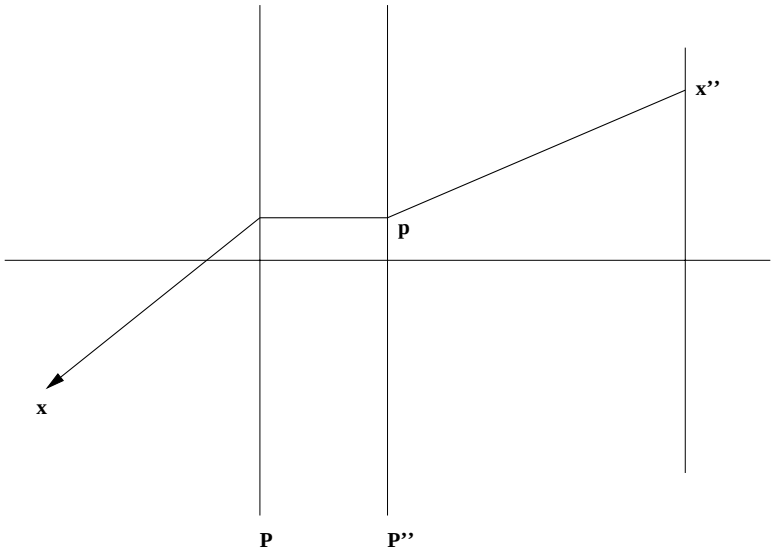


Figure 10: Ray tracing with thick lens approximation

Cook's distribution raytracing may be extended to use a thick lens approximation[KMH95] instead of a thin lens approximation. Figure 10 illustrates the procedure for using a thick lens approximation with distribution ray tracing. The thick lens image formation principles discussed earlier are put to use here. To trace a ray from an image point x'', a point p on the lens (represented by a pair of principal planes) is first chosen. Then the intersection point with the principal plane P'' is translated parallel to the optical axis. The ray is then directed towards the image, or conjugate point x, of x''. Ray tracing then proceeds in the usual manner.

The problem of analytically finding principal plane representations for combinations of thick lenses is one of considerable difficulty [JW57]. Here ray tracing comes to the rescue. As illustrated in Figure 9, paraxial rays may be used to determine the location of the principal planes. One simply has to compute the intersection of the incident and emergent rays for this purpose.

The computation of successive refractions brings up the problem of numerical precision. In brute force full simulations, this could prove to be problematic for elaborate lens systems. In the case of determining the principal planes as described above, I used multiple sampling rays, and arrived at the result using a weighted average. Rays closer to the optical axis were assigned larger weights.

3.8 Lens Aberration

The principles of optics and formulae discussed so far arise from *Gaussian optics*, which assumes that paraxial rays incident on a lens are brought to a perfect focus. In reality, a wide beam of paraxial rays incident on a lens is not brought to focus at a unique point. The resulting defect is known as *spherical aberration*. Spherical aberration is measured as the distance by which the focus is displaced along the optical axis (Figure 11). The deviation of lens behaviour from Gaussian optics also gives rise to other types of lens aberration such as *coma*, and *astigmatism* [JW57].

I use results from lens aberration theory to model spherical aberration for thin lenses. We have

$$L_s = \frac{h^2}{8f^3}\frac{1}{n(n-1)}\left[\frac{n+2}{n-1}q^2 + 4(n+1)pq + (3n+2)(n-1)p^2 + \frac{n^3}{n-1}\right] \tag{11}$$

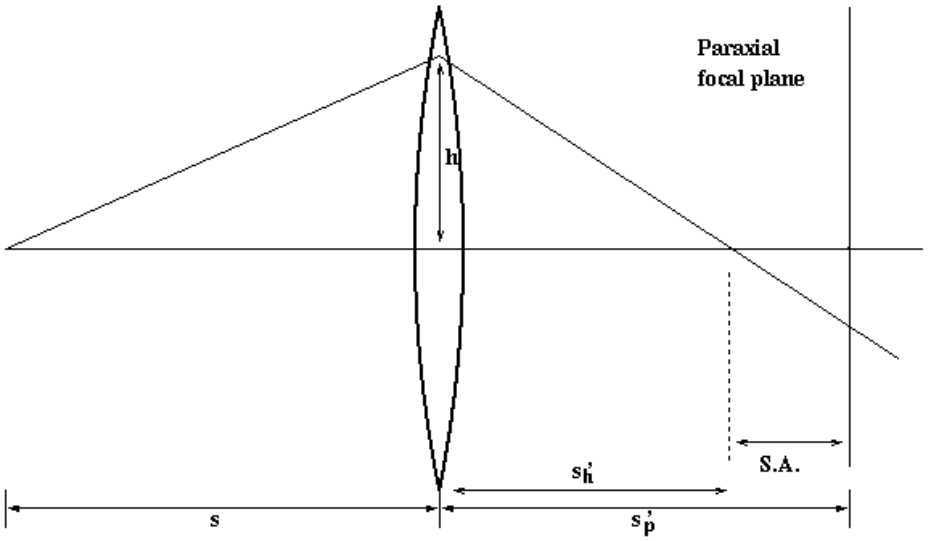


Figure 11: Spherical aberration for a thin lens

where n is the refractive index, h is the height of the ray from the optical axis, p is the *position factor* defined as

$$p = \frac{s' - s}{s' + s} \tag{12}$$

s and s' representing object and image distances respectively. The *shape factor* q is defined as

$$q = \frac{r_2 + r_1}{r_2 - r_1} \tag{13}$$

where r_1 and r_2 are the lens radii. The distance s'_h is the image distance for oblique rays incident on the lens at a height h from the optical axis, and s'_p is the image distance for paraxial rays. The sought after spherical aberration is then given by:

$$S.A. = s'_p s'_h L_s \tag{14}$$

The distance s'_p is given by

$$\frac{n}{s} + \frac{n'}{s'_p} = \frac{n' - n}{r} \tag{15}$$

The distance s'_h is given by

$$s'_h = \frac{s'_p}{1 + s'_p L_s} \tag{16}$$

We can thus incorporate a spherical aberration adjustment while tracing the ray. The treatment for thick lenses is less tractable. However, lens systems consisting of multiple thick lenses can be designed so as to reduce spherical aberration. Aberrations are more pronounced in simple lens systems (the human eye for intance).

3.9 Sampling

Owing to the discrete nature of pixels, computer graphics is inherently a sampling process. This is especially true in the case of ray tracing, where the 3D world is sampled discretely at each pixel. An image can be

thought of as a set of discrete samples of a signal representing intensity variations over space. The notion of *spatial frequency* deals with the variation of the signal over space. Since the sampling is discrete, arbitrarily fine detail in the original continuous spatial signal cannot be completely represented by the samples. This loss of information is termed *aliasing*. The pixel resolution dictates an upper bound for spatial frequencies that can be displayed. This upper bound is the *Nyquist limit*, one cycle per two pixels. Frequencies higher than the Nyquist limit cause aliasing. When a uniform sampling grid is used, these high frequencies appear as objectionable artifacts. If the sample grid is nonuniform, the high frequencies may appear as noise, which is less objectionable than aliasing artifacts [Yel83]. Techniques for nonuniform *stochastic* (probabilistic) sampling are described below. This is followed by a discussion on *reconstruction filtering*, which deals with the computation of a final value using the multiple samples obtained.

3.9.1 Jittering a Uniform Grid

This is a straightforward technique that is easy to implement, and is widely used [Wat92, Gla89, Wil94b]. A pixel is subdivided into a regular grid of subpixels. The centre of each subpixel is *jittered*, or displaced by a random offset. The jittered centre of the subpixel is then used as a sampling location, and the colour for the sample point is computed in the usual manner.

3.9.2 Poisson Disc Sampling

A more sophisticated technique that achieves superior antialiasing is *Poisson Disc Sampling*. In this technique, the sampling pattern is random, but with the constraint that no two sample points may be closer than a certain minimum distance (each point is treated as an inviolable disc) [Rip77, Mit87]. This point distribution has the characteristic of concentrating aliasing noise energy in the higher frequencies, thus facilitating its removal through *low pass filtering* [Mit91]. This concept is illustrated in Figure 12. The ω_x and ω_y axes denote the axes in the 2D frequency domain. The spectrum F of the sampling pattern S will be in the form of a spike at the origin surrounded by a noisy distribution. The convolution of the image spectrum F with S, $F * S$, yields a copy of the true image spectrum (the symmetric shape at the centre of the figure) surrounded by a halo of noise energy (scattered dots in the figure). The low pass filter (the dashed box) attenuates energy outside its bounds. It is convenient when the noise energy is concentrated in the high frequencies, thereby allowing it to be removed by the low pass filter. The Poisson Disc sampling pattern, having this property, provides good antialiasing.

3.9.3 Using a Disc-shaped Sampling Region

There are two levels of discrete sampling involved in the distribution ray tracing process. As discussed above, there is the sampling associated with the image grid. There is also the problem of sampling the lens' surface. In the case of sampling the surface of a lens, one must use a disc-shaped sampling region, since the circles of confusion are just that: circles. This may be accomplished by using a suitable mapping from the unit square to the disc-shaped region. The obvious mapping is

$$r = \sqrt{u}, \theta = 2\pi v \tag{17}$$

This mapping, however, exhibits severe distortion. One requires a mapping that takes uniformly distributed points on the square to uniformly distributed points on the disc. Shirley [Shi91] suggests mapping concentric squares to concentric circles. This mapping is illustrated in Figure 13, and for one wedge of the square, is defined by:

$$x' = 2x - 1, y' = 2y - 1, r = y', \theta = \frac{x'}{y'} \tag{18}$$

An alternative mapping [KMH95] takes subrectangles $[0, x] \times [0, 1]$ to a chord on the disc with area proportional to x, as illustrated in Figure 14.

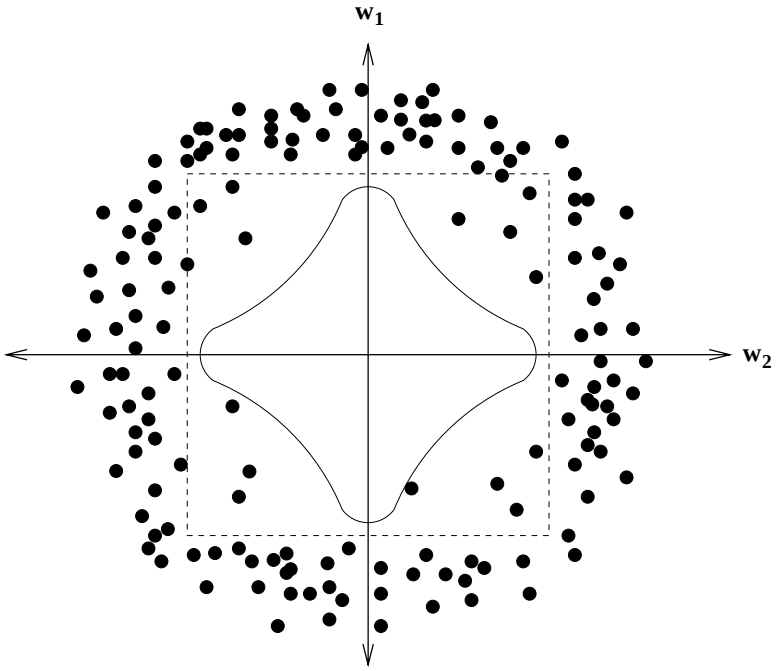


Figure 12: Spectrum of a nonuniformly sampled image.

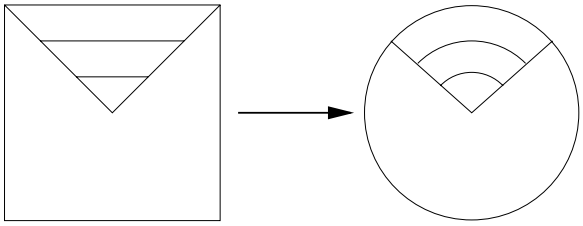


Figure 13: Mapping of concentric squares to concentric circles.

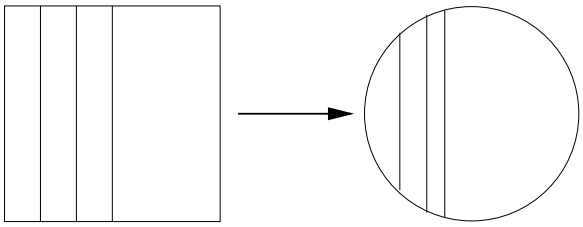


Figure 14: Mapping of subrectangles to chords of proportional area.

3.9.4 Reconstruction Filtering

The reconstruction filter interpolates samples (obtained using an appropriate supersampling pattern such as the ones described above) to recreate a continuous image (signal). This image is then low pass filtered to ensure that high frequencies are eliminated, so that aliasing will not occur while subsampling at the pixel rate (the reconstructed image is resampled using the regular pixel grid to obtain the final image). The reconstruction and low pass filters can be combined into a single filter [Mit87]. One such reconstruction

filter is the Gaussian filter given by

$$e^{-d^2} - e^{-w^2} \tag{19}$$

where d is the distance from the centre of the sampling region to the centre of the pixel, and w=1.5 is the filter width, beyond which the filter is set to zero. In the 3DCAMS ray tracing component, this reconstruction filter was applied when distribution ray tracing was used, as in the case of the thin and thick lens simulations, as these techniques inherently involve supersampling.

3.10 Results

A sample 3D scene was rendered using the following image formation models:

- pinhole model
- thin lens model
- thick lens model

The lens used for the experiments is specified in Table 2. One may compute the thin lens focal length for this lens using the *Lens Makers' Formula* [JW57]:

$$\frac{1}{f} = (n-1)\left(\frac{1}{r_1} - \frac{1}{r_2}\right) \tag{20}$$

where r_1 and r_2 are the radii of the lens surfaces from left to right, n is the refractive index of the lens medium, and f is the focal length. Substituting values from Table 2, we have

$$\frac{1}{f} = (1.1-1.0)\left(\frac{1}{80.0} - \frac{1}{-80.0}\right) \tag{21}$$

which yields $f = 400.0$ units. This value was used for pinhole ray tracing, and thin lens distribution ray tracing. The principal plane approximation for the thick lens model was determined using ray tracing, as described in Section 3.7.

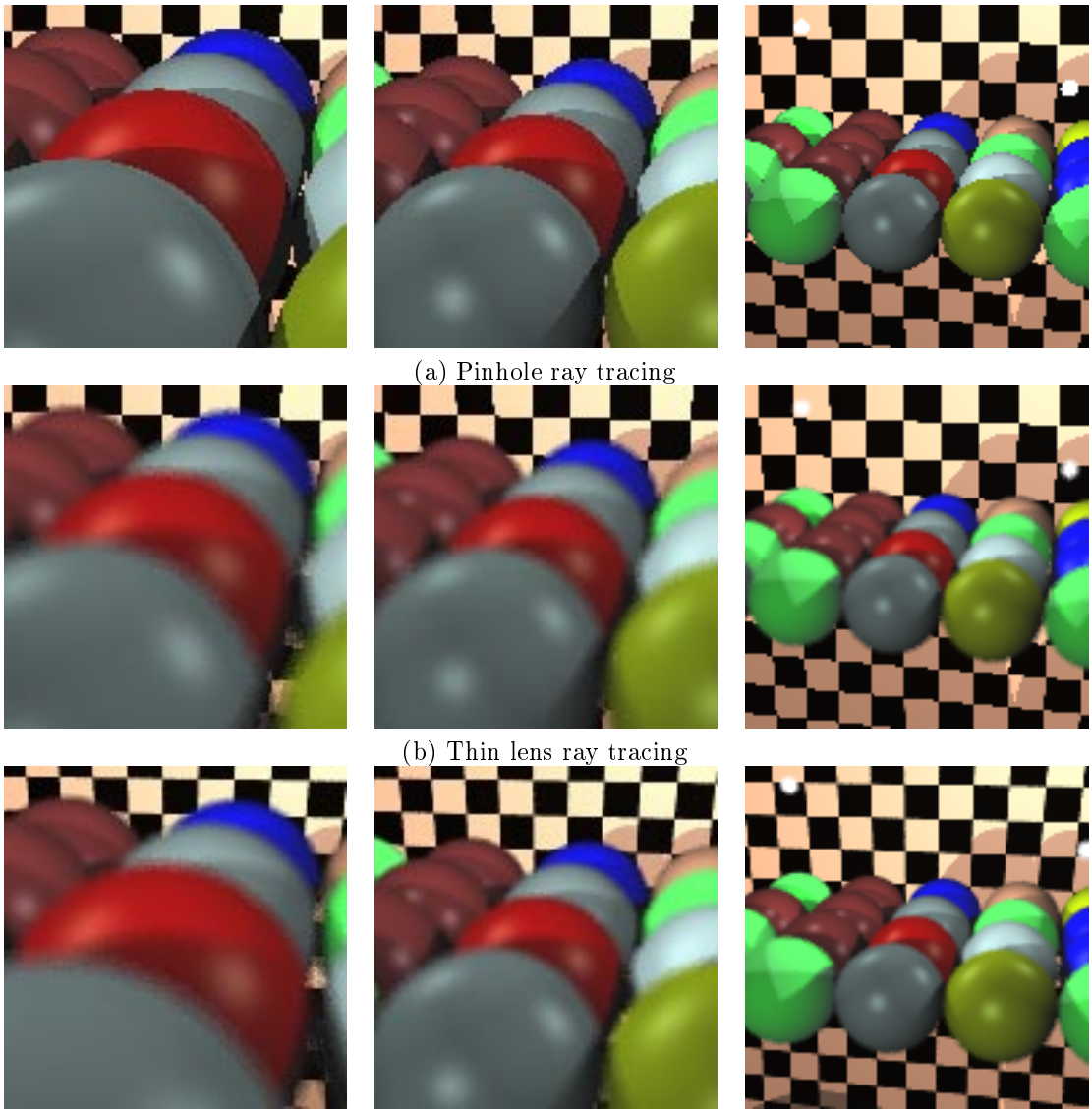
Radius	Thickness	R.I.	Aperture
80.0	6.0	1.1	50.4
-80.0	0.1	1.0	50.4

Table 2: Parameters for the lens used in the ray tracings.

The rendered images using pinhole, thin lens, and thick lens models are shown in Figure 15. An examination of the pinhole rendered images (15(a)) reveals that all objects in the scene are in sharp focus, as there is no accounting for depth of field in this model. The thin lens images (15(b)) and the thick lens images (15(c)) show the chequered background to be focussed, while the spheres in the foreground are out of focus. A comparison of these two sets of images also reveals a difference in the field of view. The thick lens images show less of the spheres in the foreground. The reason is that the field of view gets compressed due to the way the lens refracts the light rays. One would have observed the opposite in the case of wide-angle or fish-eye lenses, which increase the field of view. A point to be noted is that thick lenses introduce radial distortions, especially in the case of fish-eye lenses, and this especially affects the imaging of points distant from the centre of the field of view.

Each set of images represents views of the scene as the lens system is moved successively closer to the image plane (the leftmost image represents the position where the lens system is farthest from the image plane). This brings up another point on which the pinhole model differs from a lens system. In the case of

a pinhole camera, moving the centre of projection closer to the image plane is tantamount to decreasing the focal length. This arises from the focal length being identical to the distance of the image plane from the centre of projection. In the case of a lens system, the focal length of the system and the distance of the lens system centre from the image plane are two different quantities. The focal length of the lens system does not change (it depends on the configuration of lens surfaces used) in the examples, the distance of the lens system centre from the image plane does.



(a) Pinhole ray tracing

(b) Thin lens ray tracing

(c) Thick lens ray tracing

Figure 15: Ray tracings of a 3D scene using the different camera models

Figure 16 depicts the differences between the rendered images graphically. The differences in RGB values were averaged to yield a gray level intensity. Thus, a brighter pixel indicates greater difference. The differences between thick lens on the one hand, and pinhole and thin lens models on the other, can be seen to be very marked in the case of the background, where visibility changes appreciably as discussed above. The resulting images from thin lens and pinhole models are very similar since the field of view does not change

(a) Thick lens - Thin lens differences

(a) Thick lens - Pinhole lens differences

(a) Thin lens - Pinhole lens differences

Figure 16: Image differences between the different models (bright represents large differences)

from pinhole to thin lens (the depth of field effect alone accounts for the difference).

Figure 3.10(a) shows a model rendered using a single concavo-convex lens. Figure 3.10(b) shows the same model rendered using the double-Gauss lens. The double-Gauss lens lends a wide-angle effect i.e, serves to widen the field of view.

(a) Model rendered using concavo-convex lens

(b) Model rendered using double Gauss lens

Figure 17:

4 Camera Calibration

Camera calibration is the process of determining a camera's geometric and optical constants and it's pose in a 3D world coordinate system [Fau93, Kan93]. This process is essential for obtaining a mathematical model

of the camera's imaging process, which in turn allows one to

- project 3D points in object space to 2D points in an image.
- make 3D inferences from 2D points in an image (e.g. stereo).

There are two kinds of parameters that govern the 3D-to-2D object-image relationship: *intrinsic* and *extrinsic* parameters:

Intrinsic Parameters:

These are the camera's internal constants, and remain fixed through any motion of the camera.

- **Focal length** f: The focal length of the pinhole camera model as used in 1. This is not to be confused with the focal length specified by the manufacturer for a camera. That value usually refers to the effective focal length of the physical lens system, and is the focal length corresponding to a thick lens model. When determining the focal length by calibration, one is reducing the physical camera comprising multiple thick lenses to a pinhole abstraction and determining the focal length for the pinhole model.
- **Image centre** C_x, C_y: This parameter has multiple definitions, depending on the application context. A commonly used definition is the *numerical centre* of the image coordinates. This is valid only under an assumption of perfect imaging. Real lenses however exhibit *lens distortion*, which displaces points from their ideal projections. The point from which radial distortions originate is known as the *centre of radial distortion*. The centre of radial distortion may be recovered through calibration (Section 2.2).
- **Radial distortion coefficient** κ_1: The radial distortion coefficient determines the relationship between undistorted and distorted image coordinates:

$$X_u = X_d(1 + \kappa_1 \rho^2) \tag{22}$$

$$Y_u = Y_d(1 + \kappa_1 \rho^2) \tag{23}$$

Extrinsic Parameters:

The extrinsic or external parameters of a camera specify it's pose in the 3D world. Unlike intrinsic parameters, these change whenever the camera is moved.

- R_x, R_y, R_z (rotation)
- T_x, T_y, T_z (translation)

One may notice that focal length and image scaling cannot be uniquely determined simultaneously, although the aspect ratio is uniquely determined. For instance, focal length may be increased, and the image scale decreased, without altering the 3D-to-2D transformation. By decreasing the image scale, one means that the dimensions of the "viewing window" are increased, to maintain the same field of view as before. Since the dimensions of the viewing window in 3D space have been increased (equally in X and Y directions), pixel distances in the image now translate to greater distances on the viewing plane, i.e. image scale has been reduced. The need to fix focal length and scaling with respect to each other is where sensor parameters come into the picture:

CCD Sensor Element Parameters: These are camera constants for CCD cameras that are normally supplied by the manufacturers.

- d_x, d_y: centre-centre sensor element distance (measured in mm) in x and y directions. The sensing plane of a CCD camera consists of an array of such sensor elements. These distances are required to fix the image scale (as explained above).

- **Scaling** s_x: For CCD cameras, a scaling constant is introduced to account for the uncertainty involved in scanning. This arises due to a variety of factors, such as timing mismatches between acquisition and scanning hardware, or timing imprecision of sampling.

Sensor element parameters are usually known reliably. The information is normally supplied by manufacturers of CCD cameras. For instance, the Photometrics Star I camera [Wil95] has $d_x = d_y = 0.023$ mm/sensor-element.

Tsai's algorithm [Tsa87], calibrates a camera represented by this model, sometimes referred to as Tsai's camera model [Wil94a]. The following section describes the procedure for recovering these parameters from 3D-2D object-image point correspondences.

4.1 Tsai's Calibration Technique

4.1.1 The Imaging Model

The camera geometry for Tsai's technique is illustrated in Figure 18. (x_w, y_w, z_w) is the 3D coordinate of the object point P in the 3D world coordinate system. (x, y, z) is the 3D coordinate of the object point P in the 3D camera coordinate system, which is centered at O, the optical centre, with the z axis coinciding with the optical axis. (X_u, Y_u) is the image coordinate of (x, y, z) obtained using regular perspective projection, without distortion. (X_d, Y_d) is the actual image coordinate obtained taking into account radial distortion. The determination of the image coordinate from the 3D world coordinate is carried out in the following manner:

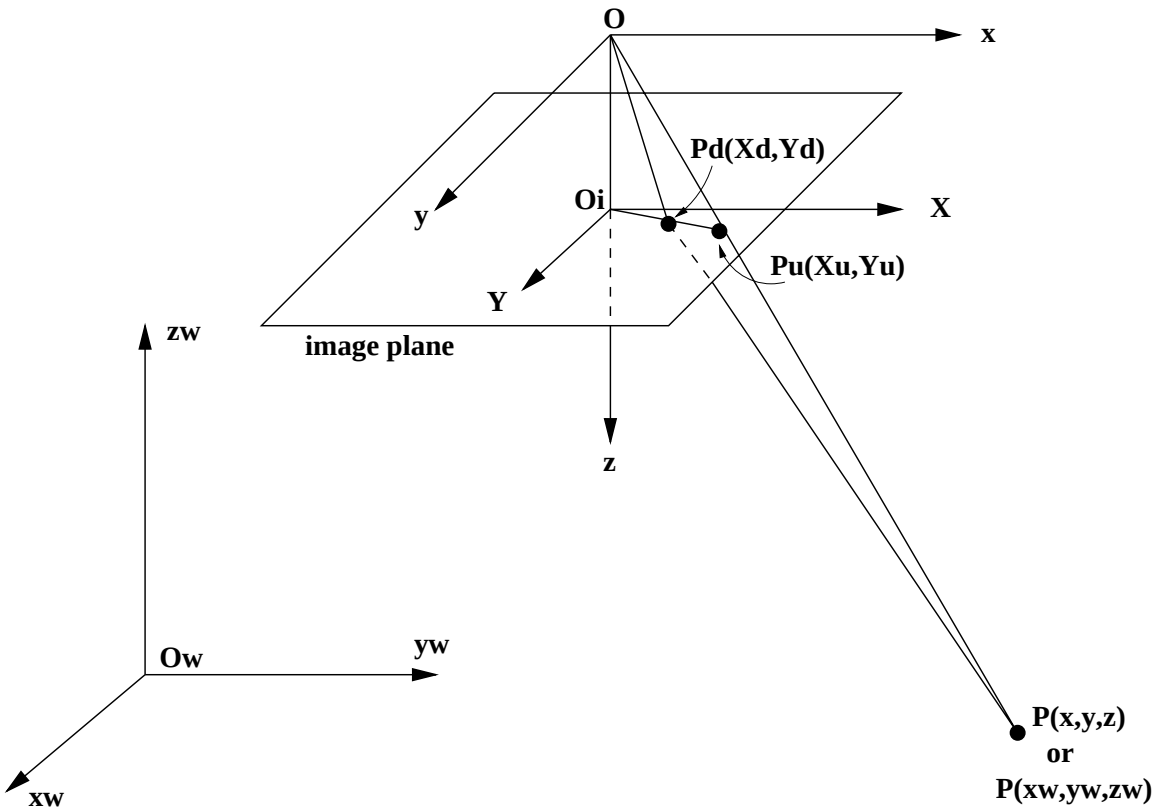


Figure 18: Tsai's camera geometry showing radial distortion

1. Rigid transformation from world coordinate system (x_w, y_w, z_w) to camera coordinate system (x, y, z) :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \tag{24}$$

where R is the rotation matrix

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{25}$$

and T is the translation vector

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{26}$$

The parameters to be calibrated are R and T.

2. Transformation from 3D camera coordinate system (x, y, z) to undistorted (ideal) image coordinate (X_u, Y_u) using perspective projection.

$$X_u = f\frac{x}{z} \tag{27}$$

$$Y_u = f\frac{y}{z} \tag{28}$$

The parameter to be calibrated is the effective focal length f.

3. Transformation from undistorted image coordinate (X_u, Y_u) to distorted (actual) image coordinate (X_d, Y_d) is given by

$$X_d + D_x = X_u \tag{29}$$

$$Y_d + D_y = Y_u \tag{30}$$

where

$$D_x = X_d \kappa_1 r^2 \tag{31}$$

$$D_y = Y_d \kappa_1 r^2 \tag{32}$$

where

$$r = \sqrt{X_d^2 + Y_d^2} \tag{33}$$

The parameter to be calibrated is κ_1.

4. Distorted image coordinate (X_d, Y_d) to computer image coordinate (X_f, Y_f) transformation.

$$X_f = s_x \frac{X_d}{d_x} + C_x \tag{34}$$

$$Y_f = \frac{Y_d}{d_y} + C_y \tag{35}$$

where

- (X_f, Y_f) are row and column numbers of the pixel in the digital image file.
- (C_x, C_y) are row and column numbers of the centre of the image.
- d_x is the centre-centre distance between adjacent sensor elements in the X direction.
- d_y is the centre-centre distance between adjacent sensor elements in the Y direction.
- N_{cx} is the number of sensor elements in the X direction.

- N_{fx} is the number of pixels in a line as sampled by the computer.
- The interpixel distance in the X direction is given by

$$d'_x = d_x \frac{N_{cx}}{N_{fx}} \tag{36}$$

- s_x is the uncertainty image scale factor.

The parameter to be calibrated is the uncertainty image scale factor s_x. To transform between the computer image coordinate and the distorted real image coordinate, the inter-pixel distance in both row and column directions needs to be known. In CCD cameras, the inter-pixel distance in the Y direction is identical to the inter-sensor-element distance d_y. However, as mentioned earlier while describing intrinsic parameters, the additional uncertainty parameter s_x has to be introduced in the X direction.

After incorporating these factors we may then relate the computer image coordinate (X, Y) to the camera coordinate (x, y, z):

$$\frac{d'_x}{s_x} X + \frac{d'_x}{s_x} X \kappa_1 r^2 = f \frac{x}{z} \tag{37}$$

$$d'_y Y + d_y Y \kappa_1 r^2 = f \frac{y}{z} \tag{38}$$

where

$$r = \sqrt{(\frac{d'_x}{s_x} X)^2 + (d_y Y)^2} \tag{39}$$

Substituting (24) into (37) and (38), we get:

$$\frac{d'_x}{s_x} X + \frac{d'_x}{s_x} X \kappa_1 r^2 = f \frac{r_1 x_w + r_2 y_w + r_3 z_w + T_x}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{40}$$

$$d'_y Y + d_y Y \kappa_1 r^2 = f \frac{x_w r_4 + r_5 y_w + r_6 z_w + T_y}{r_7 x_w + r_8 y_w + r_9 z_w + T_z} \tag{41}$$

4.2 The Algorithm

This section summarises Tsai's algorithm from [Tsa87]. The following section deals with the *radial alignment constraint* which is the distinguishing idea of this technique. This is followed by sections on calibration from coplanar points, and calibration from noncoplanar points.

4.2.1 The Radial Alignment Constraint

Tsai's algorithm is based on the radial alignment constraint, which is explained below, with reference to Figure 19.

- Assuming that lens distortion is radial, $\overline{O_i P_d}$ extending from the origin O_i in the image plane to image point (X_d, Y_d) is radially aligned with $\overline{P_{oz} P}$ extending from the point P_{oz} on the optical axis to the object point $P(x, y, z)$. The point P_{oz} is a point on the optical axis at a distance z from the origin of the camera coordinate system. $\overline{O_i P_d}$ being parallel $\overline{P_{oz} P}$ is equivalent to the following 2D statement:

$$(X_d, Y_d) \times (x, y) = 0 \tag{42}$$

or

$$y X_d - x Y_d = 0 \tag{43}$$

Substituting for x and y from (24),(25),and (26), we have

$$X_d(r_4 x_w + r_5 y_w + r_6 z_w + T_y) = Y_d(r_1 x_w + r_2 y_w + r_3 z_w + T_y) \tag{44}$$

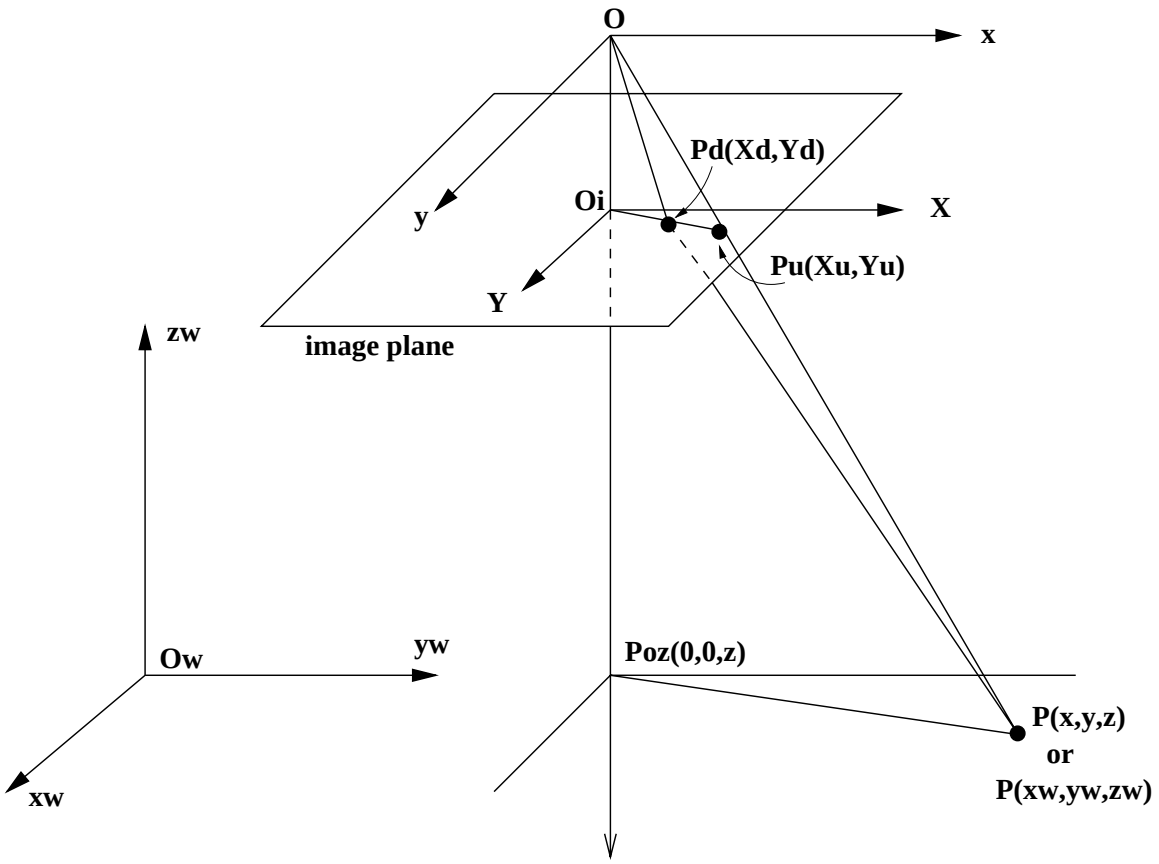


Figure 19: Illustration of the radial alignment constraint

- The direction of $\overline{O_iP_d}$ is not affected by the effective focal length f since f scales both X_d and Y_d by the same factor.
- If the world coordinate system is rotated and translated (in x and y) such that $\overline{O_iP_d}$ is parallel to $\overline{P_{oz}P}$ for every point, then the direction of $\overline{O_iP_d}$ is not affected by translation in z (z scales X_u and Y_u equally).
- The radial alignment constraint being shown to be independent of κ_1, f, and T_z, is sufficient to determine R, T_x, T_y, and s_x. A detailed proof may be found in [Tsa87].

4.2.2 Calibration Using a Coplanar Set of Points

It is convenient to use a coplanar set of data points. This arises from the ease of physically fabricating a calibration target containing coplanar data points, whose world coordinates are accurately known. The calibration procedure may be outlined as follows:

1. Compute 3D rotation, x and y translation.
   (a) Compute distorted image coordinates (X_d, Y_d) of the data points.

(b) Compute the five unknowns $\frac{r_1}{T_y}, \frac{r_2}{T_y}, \frac{T_x}{T_y}, \frac{r_4}{T_y}, \frac{r_5}{T_y}$ by solving the following linear equations:

$$\begin{bmatrix} Y_{di}x_{wi} & Y_{di}y_{wi} & -X_{di}x_{wi} & X_{di}y_{wi} \end{bmatrix} \begin{bmatrix} \frac{r_1}{T_y} \\ \frac{r_2}{T_y} \\ \frac{T_x}{T_y} \\ \frac{r_4}{T_y} \\ \frac{r_5}{T_y} \end{bmatrix} = X_{di} \tag{45}$$

for each point correspondence i with world coordinate (x_{wi}, y_{wi}, z_{wi}) and image coordinate (X_{di}, Y_{di}). With the number of points N larger than five, an over-determined linear system can be solved for the five unknowns $\frac{r_1}{T_y}, \frac{r_2}{T_y}, \frac{T_x}{T_y}, \frac{r_4}{T_y}, \frac{r_5}{T_y}$. Equation (45) is obtained from (44) by setting z_w to zero (since this is the coplanar case) and rearranging terms such that $\frac{r_1}{T_y}, \frac{r_2}{T_y}, \frac{T_x}{T_y}, \frac{r_4}{T_y}, \frac{r_5}{T_y}$ are treated as unknown variables.

(c) Compute $(r_1, ..., r_9, T_x, T_y)$ from the previously determined $(\frac{r_1}{T_y}, \frac{r_2}{T_y}, \frac{T_x}{T_y}, \frac{r_4}{T_y}, \frac{r_5}{T_y})$. Details may be found in [Tsa87].

2. Compute effective focal length, distortion coefficient, and z translation.

(a) Compute approximate f and T_z by ignoring lens distortion. By setting κ_1 to zero in (41) we get:

$$[y_i - d_y Y_i] \begin{bmatrix} f \\ T_z \end{bmatrix} \tag{46}$$

(b) Compute the exact solution with f,T_z, and κ_1 as unknowns. This is carried out by solving (41) using a standard optimisation technique with the previously computed approximate values of f and T_z as initial values.

Note: This method does not calibrate the uncertainty scale factor s_x. s_x is assumed to be known *a priori*. Hence, if s_x is known, this method may be used. Coplanar calibration is often preferable, as a calibration target consisting of coplanar points is easier to construct.

4.2.3 Calibration Using a Noncoplanar Set of Points

If the uncertainty scale factor s_x is not known *a priori*, Tsai's method for noncoplanar points should be used. The procedure is outlined as follows:

1. Compute 3D rotation, x and y translation, and the uncertainty scale factor.

(a) Compute image coordinate (X'_d, Y'_d), where (X'_d, Y'_d) is defined as is (X_d, Y_d) in (34) and (35) with s_x set to 1.

(b) Compute $\frac{s_x}{T_y}r_1, \frac{s_x}{T_y}r_2, \frac{s_x}{T_y}r_3, \frac{s_x}{T_y}T_x, \frac{r_4}{T_y}, \frac{r_5}{T_y}, \frac{r_6}{T_y}$ by solving the linear system

$$\begin{bmatrix} Y'_{di}x_{wi} & Y'_{di}y_{wi} & Y'_{di}z_{wi} & Y'_{di} & -X'_{di}x_{wi} & X'_{di}y_{wi} & -X'_{di}z_{wi} \end{bmatrix} \begin{bmatrix} \frac{s_x}{T_y}r_1 \\ \frac{s_x}{T_y}r_2 \\ \frac{s_x}{T_y}r_3 \\ \frac{s_x}{T_y}T_x \\ \frac{r_4}{T_y} \\ \frac{r_5}{T_y} \\ \frac{r_6}{T_y} \end{bmatrix} = X'_{di} \tag{47}$$

Equation (47) is obtained from (44) by rearranging terms such that $\frac{s_x}{T_y}r_1, \frac{s_x}{T_y}r_2, \frac{s_x}{T_y}r_3, \frac{s_x}{T_y}T_x, \frac{r_4}{T_y}, \frac{r_5}{T_y}, \frac{r_6}{T_y}$ are treated as unknown variables.

(c) Compute $(r_1, ..., r_9, T_x, T_y, s_x)$ from the previously determined
$\frac{s_x}{T_y} r_1, \frac{s_x}{T_y} r_2, \frac{s_x}{T_y} r_3, \frac{s_x}{T_y} T_x, \frac{r_4}{T_y}, \frac{r_5}{T_y}, \frac{r_6}{T_y}$.
The detailed derivation may be found in [Tsa87].

2. Compute effective focal length, distortion coefficient, and z translation.

   (a) Compute approximate f and T_z by ignoring lens distortion just as in the coplanar case.

   (b) Compute the exact solution of f, T_z, and κ_1 as in the coplanar case.

4.3 Calibration Data

Calibration data consists of the 3D (x, y, z) world coordinates of an object point and the corresponding coordinates (X_f, Y_f) of the feature point in the image. This section discusses some important considerations regarding obtaining and using calibration data.

As explained above, Tsai's algorithm has two variants: one for coplanar data and one for noncoplanar data. Tsai's algorithm fails if the origin of the world coordinate system is near the centre of the camera's field of view or near the Y axis of the camera coordinate system. It is easy to respect these constraints as the origin of the world coordinate system may be chosen arbitrarily (the recovered relative pose of the world coordinate system will obviously reflect the choice made).

The s_x camera parameter cannot be calibrated using coplanar data. Hence, either one should use some independent method of estimating s_x, or use noncoplanar data instead (more difficult to set up).

Basic coplanar calibration requires at least five data points. Basic noncoplanar calibration requires at least seven data points. Nonlinear optimization for these camera calibration routines is performed using a modified Levenberg-Marquardt algorithm [Wil95].

To accurately estimate radial lens distortion and image centre parameters, the calibration data should be distributed broadly across the field of view. The distribution of data points should, if possible, span the range of depths that one expects [Wil95]. To be able to separate the effects of f and T_z on the image there needs to be adequate perspective distortion in the calibration data. For adequate perspective distortion, the distance between the calibration points nearest and farthest from the camera should be on the same scale as the distance between the calibration points and the camera [Wil95]. This applies both to coplanar and noncoplanar calibration. Experiments indicate that calibration is sensitive to these data distribution factors, especially so when a modest number of data points are used.

5 User's Overview of 3DCAMS

3DCAMS is an interactive 3D visualisation system. It provides the user GUI based visualisation capability along with a range of choices in camera modelling and simulation. This section describes the functionality and use of the system.

5.1 Specifying Camera Parameters

The user may specify a known camera by loading a camera model from an ASCII specification file. The model consists of the intrinsic and extrinsic parameters described earlier. In the case of a physical lens system, the user may specify parameters for each of the constituent thick lenses as shown earlier in Table 1. Sample specifications are shown in Figure 20. For a pinhole or a thin lens, the focal length is specified. In the case of a physical thick lens system, the lens surface parameters need to be specified (the parameters following "LENSSYSTEM").

```
ROTATION
1 0 0
0 1 0
0 0 1
TRANSLATION
-50 0 20
PIERCING-POINT
0 0
WINDOW
UMIN -1.0
UMAX 1.0
VMIN -1.0
VMAX 1.0
```

```
FOCAL LENGTH (PINHOLE/THIN)
20.0

       (or)

LENSSYSTEM
NSURFACES 4
LENS 0 -80 5.0 1.5 ConvexLens1.1
LENS 6 +80 5.0 1.0 ConvexLens1.2
LENS 0 -80 5.0 1.5 ConvexLens2.1
LENS 6 +80 5.0 1.0 ConvexLens2.2
```

Figure 20: An example of camera specification

5.2 Viewing

The viewing parameters may be changed by manipulating the camera's extrinsic parameters, which may be specified as:

- Position in 3D world coordinates: This is specified as a 3D vector, and represents the translation of the camera coordinate system origin with respect to the world coordinate system.
- Orientation: This is specified as a rotation matrix, and represents the rotation of the camera coordinate system with respect to the world coordinate system.

Manipulation: The user may orient and move the camera so as to obtain a desired view of the scene.

- 3D Translation: The camera is translated in 3D space. This in effect translates the origin of the camera coordinate system.
- Orientation change: The camera may be panned, tilted or rolled.

The above parameter adjustments may be specified either as absolute quantities, or as offsets relative to the current camera coordinate system, using the mouse based GUI (to be described in a following section).

- FOV change. The camera can be manipulated to zoom in or out.

5.3 The Model

The model is specified as a topological object (to be described in detail in Section 6), consisting of faces, edges, and vertices. The pose of the model is specified as:

- Position in 3D world coordinates. This represents the translation of the model coordinate system with respect to the world coordinate system. By default, the model coordinate system origin coincides with that of the world coordinate system.
- Orientation (a rotation matrix). By default, the model coordinate system axes are aligned with those of the world coordinate system.

Manipulation: The local model coordinate system has its origin at the centroid of the model. By default, the coordinate axes are aligned with those of the world coordinate system.

- 3D Translation: This allows the user to translate the model in 3D space. This involves translating the origin of the model coordinate system accordingly.
- Rotation: Rotation may be specified with respect to model coordinate system axes in terms of pans, tilts, and rolls, or as a rotation matrix that represents the rotation of the model coordinate system with respect to the world coordinate system.

The model coordinate system is centred at the centroid of the model and is initially aligned with the camera coordinate system. It is often desirable to manipulate the model locally, in order to register it with the image, or simply to get a better view. The GUI for this capability is described in section 5.7 . At any point, the model display may be toggled on/off. Turning off the model display allows faster display of the texture mapped image (described next).

5.4 The Image

The user may load an image corresponding to the model. This image is usually obtained by photographic acquisition or by synthesis using a suitable rendering technique. For instance, one needs to load an image in order to calibrate a camera. The image may be loaded from a TIFF or PGM file (supported by the IUE). The pose of the image (considered as a planar entity in 3D space) is specified as:

- Position in 3D world coordinates: The image may be positioned as desired in 3D space. By default, the image polygon coincides with the camera's image plane, such that it covers the entire field of view.
- Orientation: Orientation may be specified as a rotation matrix that represents the rotation of the model coordinate system with respect to the world coordinate system.

Manipulation: This is achieved by representing the image as a texture mapped plane, and rendering the image polygon each time a transformation is applied.

- 3D Translation: This allows the user to translate the image in 3D space. This involves translating the origin of the image coordinate system accordingly.
- 3D Rotation: Rotation may be specified with respect to image coordinate system axes in terms of pans, tilts, and rolls, or as a rotation matrix that represents the rotation of the image coordinate system with respect to the world coordinate system.

The GUI for this capability is described in a following section. Turning off the image display allows faster display of the model.

5.5 Camera Calibration

Camera calibration requires the specification of model-image point correspondences. Both model and image thus need to be loaded to perform calibration. The image could have been acquired by a CCD camera or may have been synthesised using a physical camera model, for which the simplified pinhole abstraction is desired. An interactive camera calibration component based on Tsai's algorithm is included in the system. The calibration routines are based on freely available code [Wil95]. For basic coplanar calibration using Tsai's technique, at least 5 correspondences must be supplied. This will not allow recovery of scaling s_x. Basic noncoplanar calibration requires at least 7 point correspondences and will recover s_x.

- The user may provide model-image correspondences by pointing and clicking. Alternatively, for very precise calibration involving a large number of points, point sets may be supplied in IUE DEX format.
- A drawback to the point-and-click specification of point correspondences is the impracticality of specifying a large number of correspondences, and hence increased sensitivity to noise. To accurately estimate radial distortion and image centre parameters, the data should be broadly distributed across the field of view and range of depth.

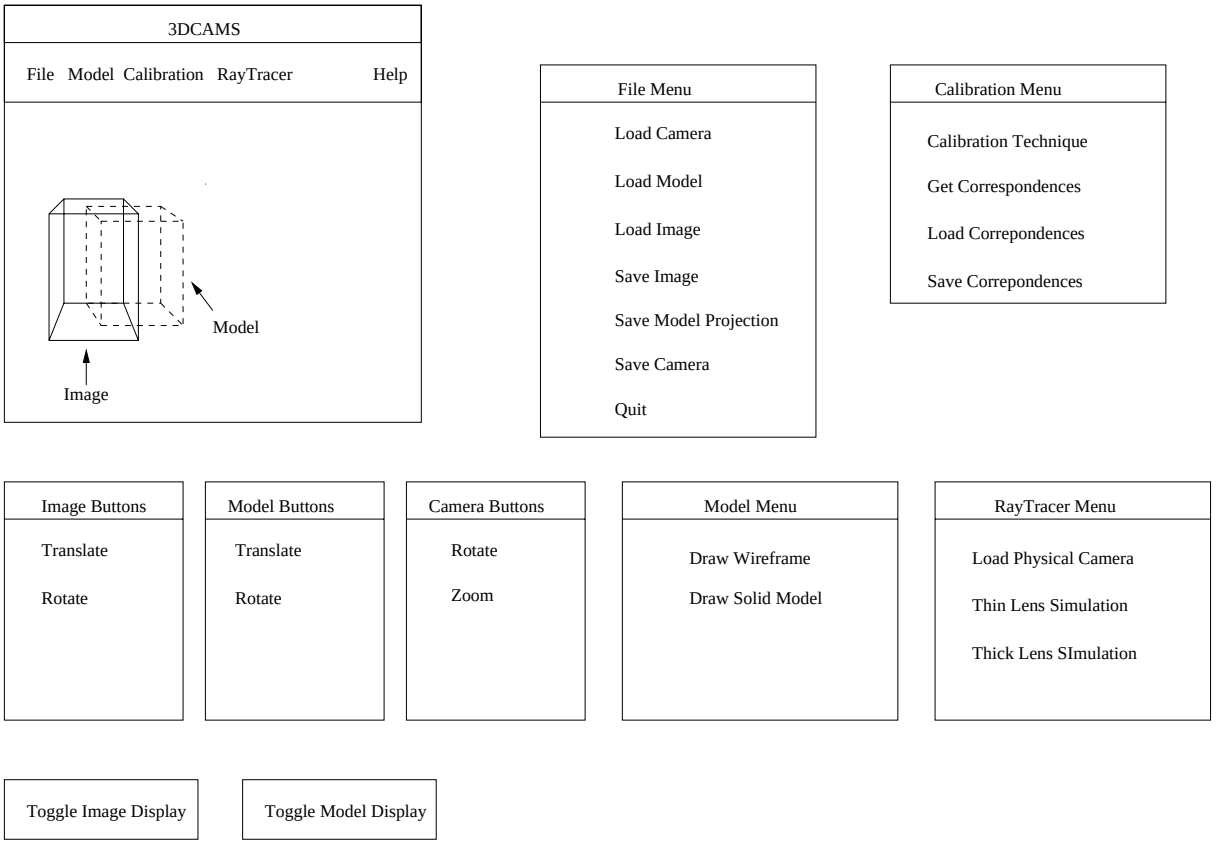


Figure 21: Schematic diagram of the visualisation system GUI

5.6 Physical Camera Simulation

The 3DCAMS system includes a physical camera simulation component. This component contains a ray tracer that implements the physical camera models discussed earlier. The thin and thick lens models are

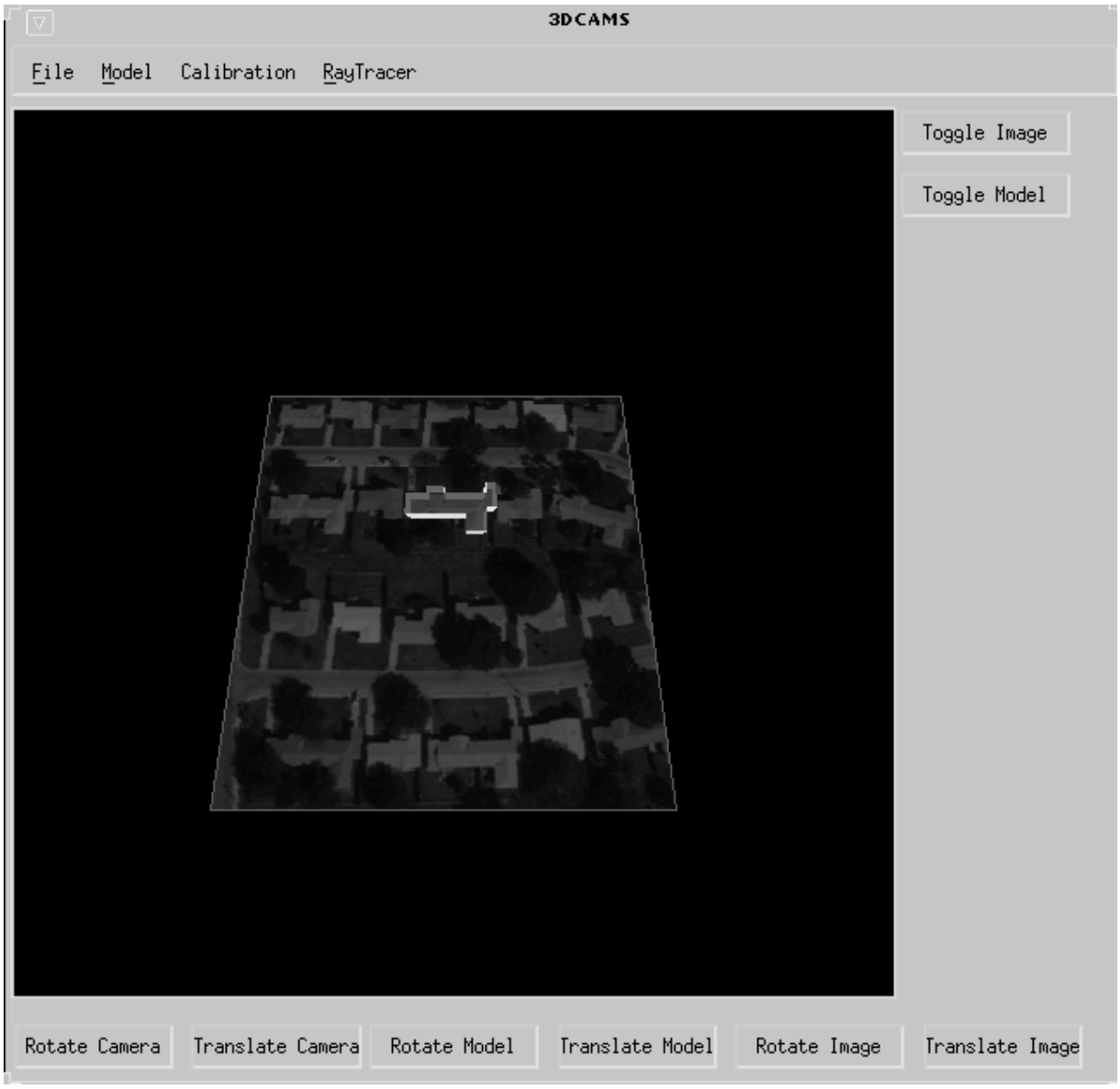


Figure 22: A snapshot of the 3DCAMS GUI

supported. Distribution ray tracing is used as discussed in detail earlier, to simulate the imaging process of these camera models. The ray tracer uses the current viewing parameters of the system's pinhole camera. The camera should be manipulated to arrive at an appropriate configuration, and then the ray tracer may be invoked to generate a high quality image.

5.7 The GUI

The GUI design is shown in Figure 21, and a sample snapshot in Figure 22. Certain menu items need explanation:

- File/Save-Model-Projection: This allows the user to save the projected vertices in IUE DEX format.

- Camera/Load: This allows the user to load a camera specification. The camera could either be a standard perspective projection camera model (as obtained by calibration for instance), or a physical lens system (as specified in Table 1 for instance). The two types of camera models have their own file formats as shown in Figure 20.
- Camera/Save: The user may save the state of the pinhole camera at any point to a file. This basically involves saving the extrinsic parameters, as intrinsic parameters do not change during camera manipulation. In other words, the new 3D translation and the new rotation matrix are changed.
- Calibration/Method: This allows the user to choose between calibrating from coplanar data points and from noncoplanar data points. The distinction between the two is explained in Section 4.3.
- Calibration/Get-Correspondences: This allows the user to establish model-image point correspondences by pointing and clicking. When multiple model vertices are perceived as candidates after the mouse-click, they are displayed in a selection box so that the user may select the desired one. The GUI for this facility is illustrated in Figure 23 and is further discussed in the following section. The user may edit previously selected correspondences to his satisfaction before invoking the calibration task.
- Calibration/Save-Correspondences: The correspondences specified through the mouse interface may be saved. This allows the user to continue later, or to refine the correspondences in multiple passes.
- Calibration/Load-Correspondences: Correspondences may be loaded from file. The user may edit these as usual, and may add more correspondences if desired.

5.7.1 Calibration Examples

Figure 23 contains an example of camera calibration using the interactive GUI. The test case contained 22 correspondences. Since the point distribution was noncoplanar, the noncoplanar calibration routine was selected. Selected model vertices are highlighted with red markers. Their corresponding image points are highlighted with green markers. Any erroneously specified correspondence could have been deleted, if desired, by removing it in the correspondence editor window. The example matches the model to its rendered image. The recovered focal length is 20.45 units against the correct value of 20 units. The recovered position of the camera is $(-49.5, 3.1, -23)$ against the correct value of $(-50, 0, -20)$. The recovered rotation matrix is a bit off from the correct matrix (identity in this example). In this example, the focal length is in error of almost 3 percent, while rotation angles are in error of almost 4 percent on average.

The calibration process is rather sensitive to noise in the data, given the low number of correspondences supplied. It is also sensitive to the spatial distribution of data points, as discussed earlier. The calibration process was repeated, with each of the data points in this sample omitted in turn. It was observed that image coordinate error due to pixel resolution contributes to the calibration error, especially in the case of points distant from the image centre. It may be worthwhile employing an appropriate feature detection algorithm to refine point selections. For instance, a corner-detection algorithm may be applied to accurately locate the feature point in the neighbourhood of the point selection. In the case of chequered or other special target patterns, one may fit lines to edges and refine the location of the feature point to subpixel accuracy using interpolation.

A second example illustrates the effect of radial distortion. In this example, the image shown in Figure 24 exhibits severe "pincushion" distortion produced by a wide-angle lens system. The figure shows the recovered radial distortion coefficient (κ_1) to be 0.000279 units/mm^2. Reprojection of model vertices using this coefficient of radial distortion indicated an average accuracy of approximately 2 image plane units.

5.7.2 Interactive Model-Image-Camera Manipulation

The poses of the model, camera, and the image(if loaded) may be loaded from parameter files by choosing the appropriate File Menu items. In addition, it is desirable to provide an on-the-fly manipulation capability. This is provided by allowing the user to switch to one of the three manipulation modes at any given instant:

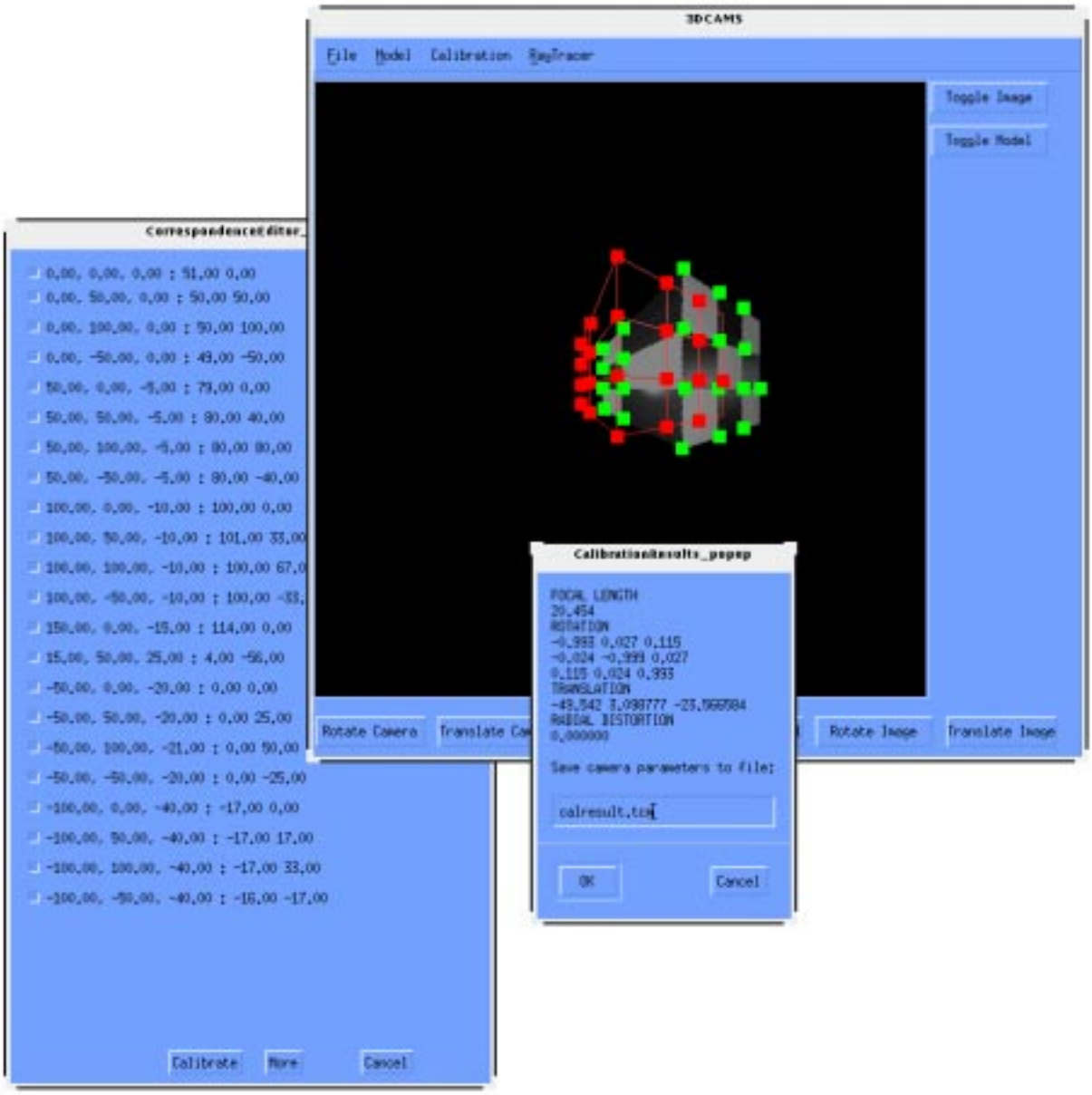


Figure 23: GUI for interactive camera calibration

- Model/Image Toggling: Both image and model display may be toggled on/off.
- Camera Manipulation: The *Rotation* button allows pan, roll, and tilt camera motion through the three mouse buttons. The *Zoom* button zooms the camera in or out based on the mouse motion.
- Model Manipulation: The *Rotation* button allows rotation about the three axes through the three mouse buttons. The *Zoom* button zooms the camera in or out based on the mouse motion.
- Image Manipulation: The *Rotation* button allows rotation about the three axes through the three mouse buttons. The *Translation* button allows translation along the three axes through the three mouse buttons. The *Zoom* button zooms the camera in or out based on the mouse motion.

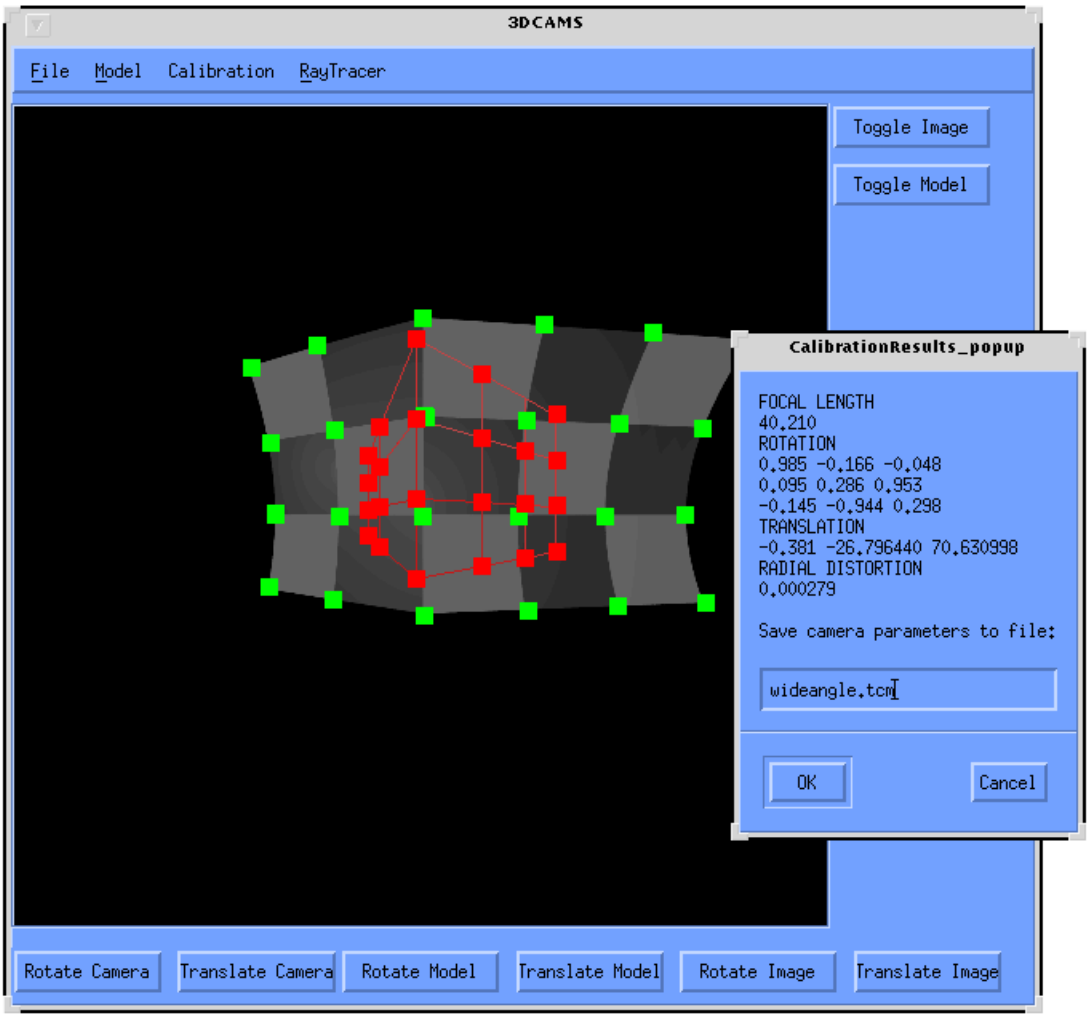


Figure 24: Calibrating for Radial Distortion

6 System Architecture and the IUE

The 3DCAMS system was developed using the IUE's C++ software development environment. The high level object oriented architecture of the system is depicted in Figure 25. The main class is the *SpatialObjectViewer* which encapsulates the GUI framework of the visualisation system. It contains instances of the key classes, the *IUETopologicalObject*, the *PinholeCamera* (Tsai's model), the *PhysicalCamera* (thin and thick lens models). Tsai's calibration techniques are implemented in the class *CalibrationRoutines*. The *SpatialObjectViewer* class is used with Motif widgets to provide the GUI for the system. The *PhysicalCamera* class encapsulates the lens simulation ray tracing routines. The classes associated with the *PhysicalCamera* class are shown in Figure 26. The 3DCAMS classes are described in greater detail in the following section. Software development in the IUE is discussed in another section.

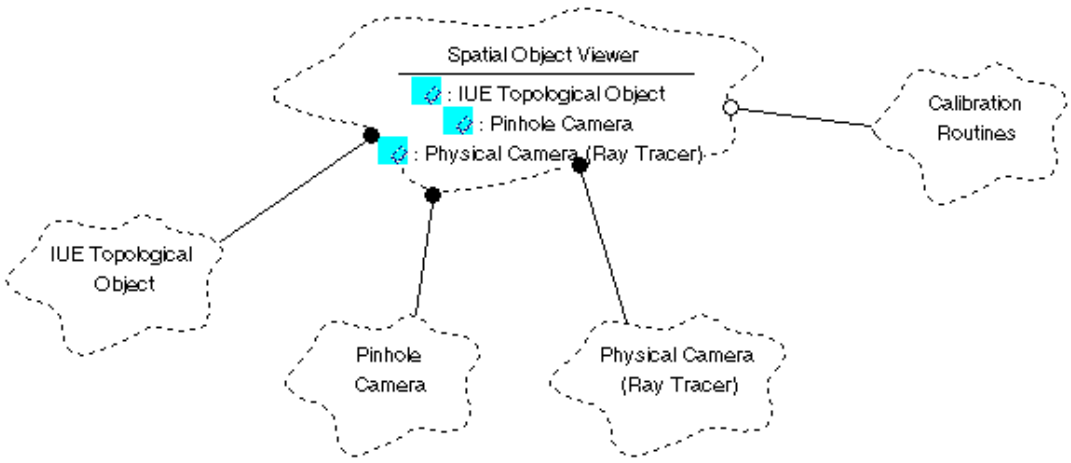


Figure 25: High level view of 3DCAMS architecture

6.1 3DCAMS Classes

This section describes the 3DCAMS classes in detail. The important class attributes and methods are listed and described. The use of various IUE classes is pointed out.

- **SpatialObjectViewer**

  Attributes:

  - spatial-object (*IUE_spatial_object*): The 3D model or spatial object, represented as a topological object (IUE toplogical objects are discussed in greater detail later).
  - image (*IUE_scalar_image_2d*): An image that contains a projection of the model or spatial object. For instance, in the Fort Hood buildings example (Figure 22), the image represents the "ground" plane upon which the buildings may be thought of as standing.
  - image-plane (*IUE_planar_face_3d*): This is the polygon onto which the image is texture mapped after performing the necessary transformations.
  - camera (*Camera*): The pinhole camera with radial distortion, to be described in greater detail.
  - raytracer (*RayTracer*): Ray tracer for the physical camera simulation, to be described in greater detail.
  - model-transform (*IUE_matrix_transform*): Represents the current pose of the model.
  - image-transform (*IUE_matrix_transform*): Represents the current pose of the texture mapped image polygon.

  Methods:

  - LoadModel: Constructs an IUE topological object from a file in DEX format.
  - LoadFredFile: Constructs an IUE topological object from a file in FRED format. The FRED format follows a topological hierarchy similar to IUE topology, and is currently used by some IUE users.

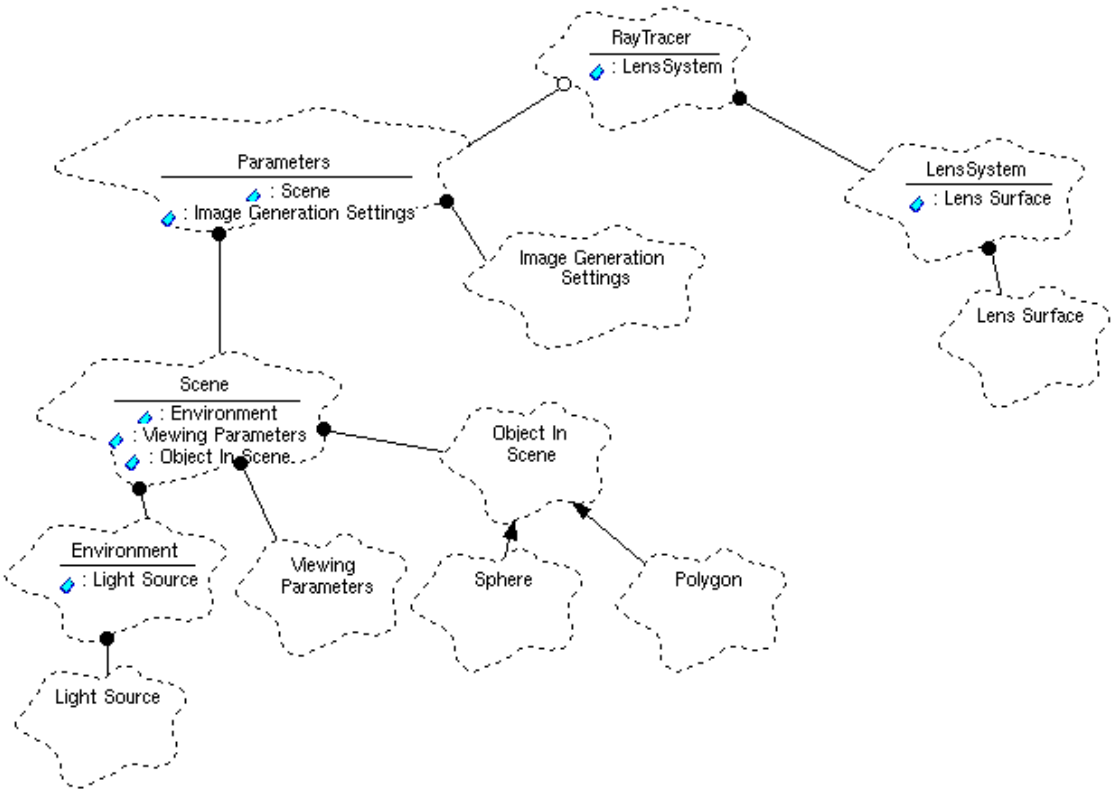


Figure 26: Design of the physical camera simulation component

- LoadImage: Reads a TIFF image file and sets up a texture mapped polygon. The IUE_TIFF_imagefiletype class method is used for this purpose.
- LoadCamera: Initialises the camera from a file that specifies parameters for a pinhole camera with radial distortion.
- DrawModel: Displays the model in shaded or wireframe mode, using OpenGL. The texture mapped image polygon is also drawn.
- RayTraceModel: Ray trace model using a physical camera model.
- GetCalibrationCorrespondences: Acquire model-image point correspondences using the GUI.

- **Camera**

  Attributes:

  - f: focal length of a pinhole camera as defined and described earlier.
  - kappa1: Radial distortion coefficient.
  - R (*IUE_matrix_transform*): Rotation matrix of the camera.
  - T (*IUE_vector_3d*): Translation of the camera in world coordinates.
  - Image plane extents: These define the viewing window.

  Methods:

- WorldToImage: Projects a point in 3D world coordinates to a point in 2D image coordinates, using the camera transformation and radial distortion.
- SaveCamera: Saves the camera state onto a camera parameters file.

- **RayTracer (Physical Camera)**

  Attributes:

  - rtparameters: Encapsulates image generation settings, including camera model mode (pinhole, thin, or thick), image resolution, sampling options.
  - scene: Encapsulates the viewing parameters, illumination settings, and the spatial objects.
  - lens-system: Implements a representation of a physical lens system, with lens surfaces, corresponding refractive indices, and class methods for performing ray-lens interaction computations.

  Methods:

  - RayTrace: Recursive ray tracing routine that samples the scene using the selected camera model and image generation settings, generates a 24-bit image that is returned as a pointer to an appropriately sized block of unsigned bytes.

- **Calibration Routines**

  Attributes:

  - correspondences: The set of model-image point correspondences that serve as input to the calibration routine. The data structure used is an *IUE_array_sequence* of *IUE_tuple_2<IUE_vertex_3d, IUE_point_2d>*.

  Methods:

  - coplanar-calibration: Implements coplanar calibration using Tsai's algorithm as discussed earlier.
  - noncoplanar-calibration: Implements noncoplanar calibration using Tsai's algorithm as discussed earlier.

6.2 The Image Understanding Environment (IUE)

The Image Understanding Environment (IUE) is DARPA's object oriented software development environment specifically designed for research in image understanding (IU) [Mea92, BDHR91, BDHR91, KLH+94, KHL95, Joh96]. It's main purpose is the facilitation of sharing of research amongst the IU community. The IUE aims to achieve this goal by providing a standard set of C++ libraries for IU applications, and a standard data exchange mechanism. By providing standard implementations for basic IU algorithms, the IUE will both facilitate rapid prototyping of IU algorithms, sharing of new IU software, and comparison amongst techniques. A standard data exchange format (DEX) has been designed with a view to facilitate exchange of research results within the IUE, as well as in non-IUE environments. The DEX standard is meant to be usable over a wide variety of systems.

6.2.1 The IUE Libraries

The IUE libraries are meant to facilitate image understanding software development. They contain basic data structures, classes suited for IU representations, and algorithm libraries. The libraries are split according to functionality, the key libraries being listed below:

- base: basic data structures (sets, arrays, STL-compliant containers etc.).
- coordsys: coordinate systems and transforms.

- image: image access.
- image-feature: image features (edgels, curves, regions etc.).
- spatial-object: spatial objects (vertices, line segments, edges, faces, other topological objects etc.)
- stat: statistics utilities.
- dex: data exchange utilities.

6.2.2 IUE Topological Objects

3DCAMS makes use of IUE's topological objects to represent 3D models. The topological objects are a part of the IUE's spatial object library. The methods defined in the IUE topology object hierarchy include vertices, zero-chains, edges, one-chains, faces, two-chains, and blocks. These are described below:

- IUE_point: An n-tuple representing a point in n-space. (e.g. IUE_point_3d represents a point (x, y, z) in 3-space).
- IUE_vertex: An IUE_vertex encapsulates an IUE_point with associated topological connectivity to its superiors, IUE_zero_chains or IUE_edges.
- IUE_zero_chain: A sequence of vertices. A zero-chain represents the boundary of an edge.
- IUE_edge: An IUE_edge represents a curve with a boundary. The curve must not intersect itself or other curves except at an IUE_vertex. The boundary of an edge is typically two vertices. This pair of vertices can be represented as an IUE_zero_chain.
- IUE_one_chain: A connected sequence of edges. Essentially a composite curve which is constructed from a set of curve segments. A closed one-chain is often called a one-cycle or edge-loop. A set of closed one-chains acts as the boundary of a surface.
- IUE_face: A connected surface region. The 1-cycles (non-self-intersecting cycles) bounding the face are traversed in a direction such that the interior of the face is always to the left. This orientation determines the direction of traversal for the edges in the boundary. The positive sense of the surface normal to the face is defined by the sense of traversal of the one-cycle boundaries. As one walks along the boundary with the enclosed surface on the left, the surface normal points upward towards one's head. As used in 3DCAMS, a face's boundary is represented by an IUE_one_chain, and all faces are assumed to be planar; they are treated as polygons.
- IUE_two_chain: A connected sequence of faces. The faces are assigned a direction symbol depending on the sense of the normal. The normal is positive if it is aligned according to the right hand rule defined by the direction of traversal of the face boundary, and negative if the normal points in the opposite direction. If an IUE_two_chain is closed it defines a volume of space and all of the surface normals must point outward from the volume. This convention is essential to support occlusion testing for ray tracing and other visibility applications.
- IUE_block: An enclosed volume of space. The volume is bounded by a set of two-cycles, i.e. closed 2-chains. The block is the central boundary representation for solids in the IUE. As used in 3DCAMS, the boundary of a block is an IUE_two_chain.

Exhaustive descriptions and specifications may be found at
http://www.aai.com/AAI/IUE/spec/.

6.2.3 Data Exchange (DEX)

The IUE's DEX standard aims to facilitate exchange of research results amongst the IU community. In this context, the DEX file format has been designed to be widely compatible, portable, and extensible. The file format is character-based, having a Lisp-like syntax, which facilitates portability, and human readability. Although the use of a character-based format reduces storage efficiency, this is not too big a hindrance when considering geometric and relational data. Image files are exchanged as external files and are merely referenced through DEX.

In order to use DEX within the IUE, users are provided with an object-oriented API that facilitates reading and writing of DEX files. Since non-IUE systems are also expected to be able to use DEX, a generic API is provided by the IUE to allow non-IUE users to develop their own DEX readers and writers. The Colorado State University vision research group for instance, comprising both IUE and non-IUE users, has developed DEX readers and writers for its data exchange requirements. 3DCAMS, being an application developed within the IUE, uses the IUE's object-oriented DEX API for I/O involving IUE spatial objects.

6.2.4 Software Development in the IUE

As mentioned earlier, the IUE provides a very rich class hierarchy for imaging applications. Classes are available for representing most entities that one deals with in imaging research. The IUE class libraries also comes with reliable HTML documentation that the IUE's class specification mechanism automatically generates. Class interfaces are consistent, predictable, and easy to use.

While the richness of the IUE class libraries is a great strength, the size of the class libraries and the template instantiation mechanism currently imposes a significant overhead on the software development cycle. Compile-link times are excessively long, often precluding rapid prototyping, which is one of the IUE's goals. In addition, many key areas of the IUE, such as the GUI API and the sensor hierarchy, are in a state of flux, or not implemented at all. For instance, the deprecation of the entire sensor class hierarchy precluded the use of DEX camera formats in 3DCAMS, or the incorporation of the ray tracing classes into the IUE hierarchy, as originally intended. While the high level of abstraction provided by IUE data structures is commendable, it is often the case that transferring information from an abstract high level IUE object to more primitive data structures (as the OpenGL API uses for instance) proves awkward. This is undesirable, since it is not realistic to hope that in the near future the IUE will provide sufficient indigenous support for graphics as to obviate the use of external libraries and APIs. Such problems presently undermine the usability of the IUE.

References

[App68] Arthur Appel. Some techniques for shading machine rendering of solids. *SJCC*, pages 37–45, 1968.

[BDHR91] J. Ross Beveridge, Bruce A. Draper, Al Hanson, and Ed Riseman. Issues Central to a Useful Image Understanding Environment. In *The 20th AIPR Workshop*, pages 21 – 30, McLean, VA, October 1991.

[BW64] Max Born and Emil Wolf. *Principles of Optics*. MacMillan, 1964.

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, jan 1986.

[CPC84] R.L. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. *Computer Graphics*, 18:137–145, 1984.

[Fau93] O.D. Faugeras. *Three-Dimensional Computer Vision*. MIT Press, Cambridge, MA, 1993.

[FD82] J. D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics*. The Systems Programming Series. Addison–Wesley, Reading, Massachusetts, 1982.

[Gla89] Andrew S. Glassner. *An Introduction to Ray Tracing*. Academic Press, 1989.

[Joh96] John Dolan and Charlie Kohl and Richard Lerner and Joseph Mundy and Terrance Boult and J. Ross Beveridge. Solving Diverse Image Understanding Problems Using the Image Understanding Environment. In *Proceedings: Image Understanding Workshop*, pages 1481 – 1504, Los Altos, CA, February 1996. ARPA, Morgan Kaufmann.

[JW57] F.A. Jenkins and H.E. White. *Fundamentals of Optics*. McGraw-Hill, 1957.

[Kan93] Kenichi Kanatani. *Geometric Computation for Machine Vision*. Oxford Science Publications, 1993.

[KHL95] C. Kohl, J.J. Hunter, and C. Loiselle. Towards a unified image understanding environment. *Proc. 5th International Conference on Computer Vision*, 1995.

[KLH+94] C. Kohl, R. Lerner, A. Hough, C. Loiselle, J.Dolan, M. Friedman, and M. Roubentchik. A stellar application of the image understanding environment: Solar feature extraction. *Proc. ARPA IUW*, 1994.

[KMH95] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. *Computer Graphics*, 29:317–324, 1995.

[Lai91] M. Laikin. *Lens Design*. Marcel Dekker, 1991.

[Mea92] J. Mundy and et. al. The image understanding environments program. In *Proceedings: IEEE 1992 Conference on Computer Vision and Pattern Recognition*, pages 185 – 214, San Mateo, CA, January 1992. Morgan Kaufmann.

[Mit87] D.P. Mitchell. Generating antialiased images at low sampling densities. *Computer Graphics*, 21:65–72, 1987.

[Mit91] D.P. Mitchell. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics*, 25:157–164, 1991.

[PC81] M. Potmesil and I. Chakravarthy. A lens and aperture camera model for synthetic image generation. *Computer Graphics*, 15:297–305, 1981.

[Rip77] B.D. Ripley. Modeling spatial patterns. *Journal of the Royal Statistical Society.*, B-39:172–212, 1977.

[Shi91] P. Shirley. Discrepancy as a measure for sample distributions. *Eurographics*, pages 183–193, 1991.

[Smi92] W.J. Smith. *Modern Lens Design.* McGraw Hill, 1992.

[STH80] C. Slama, C. Theurer, and S. Henriksen. *Manual of Photogrammetry, Fourth Edition.* American Society of Photogrammetry, 1980.

[Tsa87] R.Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 4:323–344, 1987.

[Wat92] C.D. Watkins. *Photorealism and Ray Tracing in C.* M and T, 1992.

[Wil94a] R.G. Wilson. *Modeling and Calibration of Automated Zoom Lenses.* PhD thesis, Carnegie Mellon Uniiversity, 1994.

[Wil94b] Nicholas Wilt. *Object Oriented Ray Tracing.* John Wiley and Sons, 1994.

[Wil95] R.G. Willson. *Freeware Implementation of Roger Tsai's Calibration Algorithm.* http://www.cs.cmu.edu/People/rgw/TsaiCode.html, 1995.

[Yel83] J.I. Yellott. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221:382–385, 1983.