# Technical Memorandum


20 Nov 2007

**To:** WaveTrain Users

**From:** Robert W. Praus II, Keith Beardmore, Boris P. Venet
MZA Associates Corp.

**Re:** The "Light Tunneling" Optical Propagation Method in WaveTrain

**Abstract**

We have implemented an approximate procedure for physical-optics propagation, dubbed "Light Tunneling", within MZA's WaveTrain optical propagation code suite. Motivations for the procedure are to speed up execution time for simulating the dynamic imaging through turbulence of extended, optically-rough scattering scenes, in particular when the illumination has short temporal coherence length. The procedure is based on the physical propagation of a mesh of individual point sources, and a special interpolation procedure is used to interpolate between the physically-propagated point-spread functions. We describe the concept in detail, and we give usage instructions for the WaveTrain Light Tunneling modules. We also review the issues that govern the validity of the Light Tunneling procedure.

# The "Light Tunneling" Optical Propagation Method in WaveTrain

*Robert W. Praus II, Keith Beardmore and Boris P. Venet*
*MZA Associates Corp.*
*20 Nov 2007*

## 1 Introduction

The "Light Tunneling" method is an approximate procedure for the numerical simulation of physical-optics propagation from extended, optically-rough reflecting (scattering) scenes. The model would apply to self-emission from extended scenes as well. The procedure accounts for distributed atmospheric turbulence in the path from reflector to imager, and the procedure allows simulation of the image temporal dynamics. The procedure is particularly oriented to the treatment of highly anisoplanatic turbulence geometries. The procedure is most valid for illumination having relatively short temporal coherence length (relatively broad optical spectrum). Certain elements of the approximation procedure are designed to significantly speed up the numerical computation, by comparison with previous diffractive propagator approaches. The Light Tunneling method has been implemented in a set of new library modules in MZA's "WaveTrain" optical propagation code.

Consider the sketch in Figure 1. An unspecified source illuminates a scene which acts as an optically rough reflector. Light is scattered in accordance, fundamentally, with the BRDF (Bi-directional Reflectance Distribution Function) map of the reflecting surface. The reflected (scattered) light propagates through dynamic turbulence to an imaging system, and we wish to compute the irradiance in the image plane conjugate to the scattering surface. In the Light Tunneling method, we explicitly model the light emanating from areal

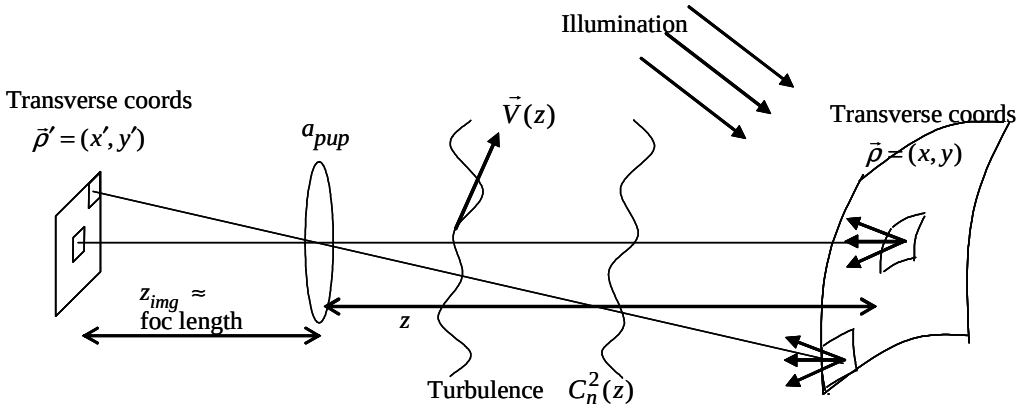


**Figure 1: Imaging through turbulence of an extended reflecting (scattering) scene**

patches of the reflector as a set of point sources, which are temporally incoherent relative to each other. The beams from each point source to the plane of the imager aperture are propagated separately (on separate

numerical grids) according to standard wave-optics procedures. That is, the turbulence is represented by a set of phase screens, and each (initially) spherical wavefront is propagated from phase screen to phase screen, using Fresnel propagation between screens. Each point-source field is then propagated from imager pupil to image plane using a single Fourier transform, and the instantaneous image plane irradiance is computed. An interpolation procedure is now defined so that (i) the conventional Fresnel propagation procedure can be restricted to a relatively sparse grid of point sources, and (ii) intensity point-spread function (PSF) results for intermediate point source locations can be estimated by the interpolation. Computational speed-up relative to previously-used diffractive propagator approaches is expected from several aspects of the new method:

(i) The interpolation concept allows us to minimize the number of point sources that need to be physically propagated.

(ii) The propagation grid for each point source can be substantially smaller than the grid required to physically propagate the entire reflected complex field. This assumes that the imaging aperture is relatively small compared to the reflecting scene, which is in practice often true.

(iii) Other WaveTrain methods for modeling short-coherence light scattering from rough surfaces require the propagation of multiple monochromatic scattering realizations, which then combine to reduce the contrast of the rough-surface speckle (but still retain the correct turbulence effects). The Light Tunneling method does not use micro-roughness realizations at all, since the method simply adds *irradiances* of all the interpolated PSFs. This precludes the formation of *any* image-plane speckle due to surface micro-roughness, and so the method does *not* describe all image details in highly monochromatic illumination. The method is intended to accurately model situations where the turbulence phase perturbations dominate the instantaneous image intensity pattern, without noticeable rough-surface speckle structure. If the temporal coherence length of the illumination is short enough, then the superposed point sources can be legitimately treated as incoherent, and there is in reality no appreciable rough-surface speckle. However, even if the illumination is fairly monochromatic, there are other mitigating factors that may allow Light Tunneling to be applied; we discuss this further below.

In Figure 1, the sketch is deliberately ambiguous about whether the model illumination *incident on the reflector* propagates through turbulence phase screens. The choice of how to treat the incident illumination will depend on the physical problem to be modeled, and this is discussed further below. At present, we just emphasize that the new WaveTrain Light Tunneling modules are *not* used to generate the light incident on the reflecting scene. An incident light field appropriate to the specific problem must be generated with other existing WaveTrain modules.

Section 2 of the present document expresses the Light Tunneling algorithm quantitatively, and gives details of the critical PSF interpolation procedure.

Section 3 discusses the domain of validity of the Light Tunneling approximations.

Section 4 briefly discusses potential modifications or extensions of the Light Tunneling procedure.

Section 5 explains the current WaveTrain implementation of Light Tunneling. This section provides the usage instructions needed to insert a Light Tunneling calculation into a WaveTrain propagation system.

Section 6 is a follow-on to Section 5, giving a numerical example of the key parameter settings and showing a sample of simulated imagery.

Section 7 explains some radiometry details of the WaveTrain implementation.

For WaveTrain usage purposes, readers need only master Sections 5 and 6 of the present document; readers can skim Sections 2 and 3 to grasp the basic concept and domain of validity of the Light Tunneling method. The radiometry implementation details in Section 7 are principally for the benefit of designers who might need to modify the Light Tunneling modules, or create analogous new material. The WaveTrain user need not follow these details to obtain radiometrically correct results.

## 2 Quantitative description of the "Light Tunneling" method

### 2.1 Basic concept

We are interested in the imaging through turbulence of an extended, optically-rough reflecting (scattering) surface that has been illuminated by some unspecified source. Figure 1 illustrates the scenario and defines some variables.

If the scene is characterized fundamentally by BRDF (the Bi-directional Reflectance Distribution Function), then we can express the irradiance (units: W/m$^2$) of a completely unaberrated ("ua"), incoherent image by the expression

$$I_{img}^{ua}(\vec{\rho}') = F \cdot I_{inc}(\vec{\rho}) \cdot B(\vec{\rho}) \ , \quad \text{where} \vec{\rho}' = M\vec{\rho} \tag{1}$$

where

$I_{img}^{ua}$ is the irradiance (W/m$^2$) in the image plane

$\vec{\rho}$ is the *object-space* position coordinate transverse to the optical axis of the imager.
We assume that all parts of the reflecting scene are within the depth of focus; this condition should be adequately obeyed in remote-sensing applications.

$M$ is the image magnification (image height/object height) for the conjugate distances in question.
In general, the sign of $M$ may be positive or negative, depending on the design of the optical system; in WaveTrain, all our imagers are just characterized by an effective (system) focal length, and we assume that $M$ is negative (inverted image).

$\vec{\rho}' = M\vec{\rho}$ is the image-plane coordinate optically conjugate to $\vec{\rho}$.

$B$ is the BRDF, which for interesting cases varies across the scene. For purposes of WaveTrain input, $B$ must be expressed as a function of the coordinate $\vec{\rho}$ transverse to the z-axis of the imager.

$I_{inc}$ is the irradiance (W/m$^2$) incident on the scattering scene.

$F$ is a radiometric factor that ensures $I_{img}^{ua}$ yields W/m$^2$ in the image plane.

Equation (1) expresses the condition of completely unaberrated imaging: that is, no path turbulence from scatterer to imager, and no aperture diffraction at the imager. Sometimes we call $I_{img}^{ua}$ the "pristine" image: it is simply a demagnified version of the (incident irradiance × reflectance) product.

Now suppose that a point source of *unit* angular intensity (i.e., 1 W/sr), located at $\vec{\rho}$, is physically propagated through the turbulence and the imager aperture, to the sensor plane of the imager. Let the image-plane irradiance pattern (units: W/m$^2$) due to this point source be designated by $S(\vec{\rho}';\vec{\rho})$. This point spread function (PSF) incorporates both the *instantaneous* turbulence and aperture diffraction. $S(\vec{\rho}';\vec{\rho})$ is concentrated around $\vec{\rho}' = M\vec{\rho}$, but it has non-zero width in the $\vec{\rho}'$ space. If we compute *S* for each point of a mesh of $\vec{\rho}$ values that span the reflecting scene, weight each *S* by the scene brightness at $\vec{\rho}$, and sum over the source points, then we will have a representation of the instantaneous, extended image as distorted by turbulence and aperture diffraction. We only sum point-spread *irradiances*, so this technique best models illumination whose spectral bandwidth is sufficient to eliminate rough-surface speckle in the return propagation path. Expressed algebraically, the image-plane irradiance at any image-plane coordinate $\vec{\rho}'$ is thus approximated by

$$I_{img}(\vec{\rho}') \approx \frac{1}{N}\sum_{i,j} S(\vec{\rho}';\vec{\rho}_{i,j})\left[I_{inc}(\vec{\rho}_{i,j})\, B(\vec{\rho}_{i,j})\,(\Delta^2\rho_{i,j})\right] \tag{2}$$

The expression in square brackets in Equation (2) is the weight factor that characterizes the strength of the individual point sources. The strength is parametrized in terms of several key proportionalities. The dependence on the $I_{inc}B$ product is obvious. The need for explicitly including the area element $\Delta^2\rho_{i,j}$ is perhaps less obvious, but the main reason is to make precise contact with continuum-space theoretical formulas, in terms of limiting operations. Finally, the factor 1/*N* will contain any remaining radiometric normalizing factors that are needed to make $I_{img}$ emerge in image-plane irradiance units (W/m$^2$): see Section 7 for details. The indices $(i, j)$ refer to the 2D mesh on which the unit-angular-intensity point sources exist. It is important to understand that the instantaneous PSF $S_{i,j}$ has a functional form that may vary significantly (due to turbulence anisoplanatism) as $(i, j)$ varies across the scene. The variation may be large or small, depending on the level of anisoplanatism that results from the problem geometry.

## 2.2 Relation to convolution

If the instantaneous point spread function $S_{i,j}$ were spatially invariant, i.e., expressible in the form

$$S(\vec{\rho}';\vec{\rho}_{i,j}) = S(\vec{\rho}' - M\vec{\rho}_{i,j};0) \tag{3}$$

then the sum in Equation (2) would be a discrete representation of a convolution. However, the general case that we want to consider in wave optics simulation has a wide enough scene so that the *instantaneous* $S_{i,j}$ is spatially variant. In that case, Equation (2) is still perfectly valid, but it *cannot* be reduced to a convolution kernel.

## 2.3 Final form of the image computation: interpolation of the point spread functions

One key motivation behind the Light Tunneling method is to speed up simulation execution time. In the introduction, we listed three elements of the procedure that tend to reduce the execution time. We now discuss one of these, namely the interpolation concept. The time required to evaluate Equation (2) evidently will depend heavily on the number of points $(i, j)$ for which we physically propagate the model point sources (using conventional screen-to-screen numerical Fresnel propagation for each point source separately). An important time-saving element of the Light Tunneling method is the idea that the PSFs can be interpolated in the following sense. Let the indices $(i, j)$ represent a relatively sparse source-point mesh, for which PSFs are computed by the standard, relatively time-intensive, method of numerical Fresnel propagation. Now we define an interpolation scheme by the following steps:

(a) We characterize each physically propagated PSF by a few of its low-order moments. In particular, we will use: (i) the centroid shifts (with respect to the unperturbed PSF centers); (ii) the x and y rms widths, and an x-y correlation factor; and (iii) the spatial integral of the PSF (the total power captured by the receiver aperture).

Note that the total power fluctuation corresponds to aperture-averaged scintillation: this may be a small effect, even in strong turbulence, if the imager aperture is large compared to the scintillation correlation length.

(b) Next, we *estimate* the PSFs due to intermediate source points that comprise a mesh significantly finer than the sparse $(i, j)$ points. Let the mesh of interpolated source points be designated by indices $(l,m)$. We begin by interpolating onto the finer $(l,m)$ grid each of the six maps computed in step (a) at the $(i, j)$ mesh points. These six maps were the x- and y-centroid displacement maps, the three 2nd-moment maps, and the power map.

(The interpolation is only useful if the function interpolation method is faster than performing physical propagations for point sources at all the interpolated mesh points!)

(c) Next, we define interpolated PSFs resulting from each $(l,m)$ source point as Gaussians whose parameters match the interpolated moments computed in step (b).

(d) Finally, we weight each interpolated Gaussian by the incident irradiance and the reflectance map at $(l,m)$, and sum all these overlapping interpolated Gaussians to generate the final irradiance image. The final image will also be defined on the $(l,m)$ mesh.

Thus, modifying Equation (2) to use all the interpolated PSFs, the final form of the instantaneous image formula is

$$\boxed{I_{img}(\vec{\rho}'_{l',m'}) \approx \frac{1}{N}\sum_{l,m} S^{gauss}(\vec{\rho}'_{l',m'};\vec{\rho}_{l,m})\left[I_{inc}(\vec{\rho}_{l,m})\,B(\vec{\rho}_{l,m})\,(\Delta^2\rho_{l,m})\right]} \qquad (4)$$

The indices $(l',m')$ refer to the same mesh as the dummy summation indices $(l,m)$. In Section 7, we will derive a simple expression for the normalization constant, *N*, to ensure that $I_{img}$ has the correct overall

radiometric scaling. We will call Equation (4) the Light Tunneling Imaging Equation. We emphasize once again that the $S^{gauss}$ corresponding to different indices $(l,m)$ are different functional shapes, so that Equation (4) is *not* a convolution.

### 2.4 Choice of meshes

The Light Tunneling method as defined above does not precisely identify the spacing required for the physically-propagated point-source mesh. To make the method practical in terms of computational time, we want to make that mesh as sparse as possible, but still dense enough so that the PSF interpolation is fairly accurate. Evidently the allowed spacing of the grid is proportional to the isoplanatic angle for the problem's turbulence geometry, which can be estimated analytically (i.e., without wave-optics simulation). The usual turbulence isoplanatic angle, denoted $\theta_0$, refers to the exact PSF. Since we are using only a Gaussian match up to second moments to do PSF interpolation, it is hoped that the allowed physically-propagated point source spacing can be substantially greater than $\theta_0$. Quantitative investigation of this issue is yet to be done.

We discuss the mesh choices further in Section 3.1 devoted to the validity of the Light Tunneling model, and in Section 5 on the WaveTrain implementation.

### 2.5 Nomenclature comments

Originally, the propagation concept described above was called "Tilt Tunneling". The "Tilt" referred to the fact that only the centroid displacement moments (tilts) and powers were originally used. The picturesque "Tunneling" refers to the way in which the shifted PSFs bleed into neighboring pixels of the pristine image. The name "Tilt Tunneling" may still be found in older MZA documents on the propagation method.

Since the second-order moments are now used in addition to the centroid shifts, MZA now refers to the method as "Light Tunneling". Perhaps a more pedantic but descriptive name would be the "Intensity-PSF Interpolation Method".

## 3 Comments on the domain of validity of the model

Light Tunneling invokes two types of approximations. (a) the shape and interpolation of the point-spread functions (PSFs), and (b) the addition of PSF intensities (i.e., incoherent addition).

### 3.1 The moments and interpolation approximation

The instantaneous PSF is characterized as described in Section 2 by a few low-order moments (x and y centroid displacements, x and y widths and correlation coefficient, and total power), and then a Gaussian shape is defined having matching moments. Admittedly, inspection of instantaneous turbulent PSFs certainly shows large deviations from a Gaussian shape. However, the motivation for the approximation is that the perturbed image of an extended scene is probably dominated by the low-order moments. Use of the moments replaces the PSF function by a small, finite set of parameters. Each parameter, e.g., the x-centroid displacement, then

constitutes a discrete function on the mesh of physically-propagated point sources, and allows an obvious spatial interpolation scheme. Once we accept the restriction to a few moments, the choice of a Gaussian shape when constructing the image sum is convenient and likely as good as any other choice. Use of the three second-order moments accounts for any low-order asymmetry in the PSF, including an arbitrary orientation of the 2-D Gaussian with respect to x and y.

A critical parameter choice that the user faces is the specification of the grid on which the physically propagated point sources reside. Supposing that the low-order moments approximation is acceptable, we must still obtain the physically-propagated moments on a sufficiently fine grid so that interpolation between them makes sense. A guideline for choosing this grid spacing is as follows. The angular separation of source points over which the PSF remains substantially constant is the isoplanatic angle, denoted $\theta_0$. A known closed-form integral estimates this number as a functional of the $C_n^2$ distribution along the propagation path. However, the standard isoplanatic angle refers to the full, exact PSF, while the low-order moments actually are associated with larger isoplanatic angles. For example, the x-centroid deviation (x-tilt) should be substantially constant over an angle larger than $\theta_0$. Therefore, the point-source grid spacing should be guided by formulas for the "relaxed" isoplanatic angles. Just how much relaxation can be tolerated for accurate results must still be investigated. As discussed somewhat further in Section 3.3, the accuracy requirements undoubtedly vary with the type of subsequent processing that is attempted with the simulated imagery.

## 3.2 The incoherent sum approximation

The second type of approximation in Light Tunneling is that we add the irradiances of the propagated and interpolated PSFs (incoherent addition of the PSFs). This has the effect of *completely* eliminating any rough-surface-speckle irradiance fluctuations in the image plane. If the illumination incident on the scattering surface is temporally sufficiently incoherent (has sufficient optical bandwidth), then this is an excellent approximation. That is, if the illumination has short temporal coherence, then the light reflected from neighboring point sources on the rough reflector would typically travel sufficiently-different path lengths that the superposition at the receiver is no longer coherent for typical sensor integration times. This limiting case was a principal motivation for the development of the Light Tunneling approach: simulating the scattering of relatively broadband illumination with previously available WaveTrain techniques was very time-intensive, because of the large number of realizations required. The latter difficulty exists in both the WaveTrain *IncoherentReflector* and *PartiallyCoherentReflector* models: in the first case, propagation is done against multiple realizations of the rough surface, while in the second case multiple realizations of illumination temporal phase are required. By contrast, the Light Tunneling method does not use a roughness realization of the surface as such. The tradeoff is that separate propagations are done for each of many point sources. (Further discussion of the modeling concepts in WaveTrain's other rough reflector modules can be found in the WaveTrain User Guide[1]).

The question of what scenarios qualify as sufficiently incoherent to make Light Tunneling useful is a rather complicated one. To form an initial impression, consider the image-formation concept sketched in Figure 2. The rough scattering surface is visualized as a dense, discrete set of scattering points, whose surface heights constitute a random process whose standard deviation is at least several wavelengths, and whose transverse correlation length is very small. Let us consider the diffraction-limited case (no turbulence). The imaging optics have a finite pupil size, $D_{pup}$, and hence a finite resolution cell. The diameter of the resolution cell in the object space is approximately $\ell_{res,O} = (\lambda z_O)/D_{pup}$, where $z_O$ is the object distance. Now at a fixed transverse position in the image plane, the net complex field is the superposition of many Airy disk

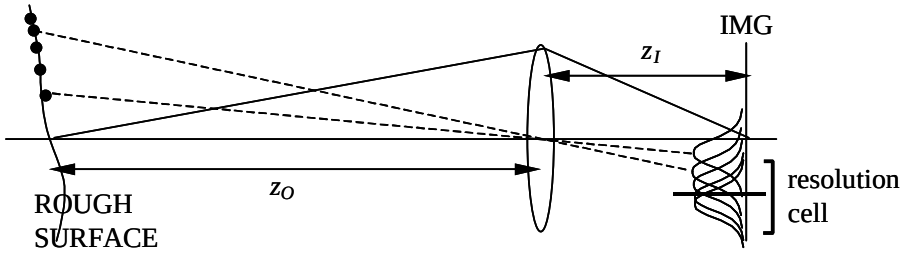


**Figure 2: Superposition of PSFs in the image plane**

cores: in fact, roughly as many as there are source points within one resolution cell. *If* the large majority of these Airy cores are temporally incoherent with each other, then the net irradiance will be essentially the sum of the individual irradiances. This condition is satisfied if, over the transverse distance $\ell_{res,O}$, the rms surface height variation, $\sigma_h$, is large compared to the illumination temporal coherence length, $\ell_{coh}$. Note in particular that any tilt or large-scale structure of the scattering object contributes to the relevant $\sigma_h$. To estimate the numbers involved, consider a simple table of $\ell_{res,O}$ values, assuming $\lambda$ = 1$\mu$m and $D_{pup}$ = 1m:

| $z_O$ | $\ell_{res,O}$ |
|---|---|
| 10 km | 1 cm |
| 40 km | 4 cm |
| 200 km | 20 cm |

**Table 1: Resolution cell width, object space, for $\lambda$ = 1$\mu$m and $D_{pup}$ = 1m.**

Now consider an illuminator coherence length $\ell_{coh}$ = 1 mm. Although many laboratory lasers have a longer coherence length, others of interest (e.g., typical high-power diode lasers) have a shorter one; furthermore, special external modulation techniques can be used to spoil the coherence of the "better" lasers if that is desirable. Now the question is, over the tabulated transverse $\ell_{res,O}$ spans, is it likely that $\sigma_h$ >> ($\ell_{coh}$ = 1 mm)? For natural scenes, particularly if viewed even slightly off normal, the condition is probably satisfied for all the cases in the table. On the other hand, if we have an illuminator with $\ell_{coh}$ = 50 cm or something in that ballpark, then the answer is no for all cases in the table. In sum, the above order-of-magnitude reasoning demonstrates that:

(a) There is definitely a practically accessible regime where the imaging behavior is sufficiently incoherent to satisfy the requirements of Light Tunneling. Of course, "sufficiently" incoherent is a relative term: one may still have, say, a few percent speckle contrast in the physical situation, but one must hope that effects of this magnitude are negligible for the application in question.

(b) There are also practical cases where the incoherence requirement is definitely not satisfied.

Users of Light Tunneling must apply the above qualitative analysis to determine whether the propagation method suits their application.

While the Light Tunneling approach is good for the incoherent limit, it lacks *any* of the rough-surface-speckle irradiance fluctuations that are physically present, whatever the contrast level may be, at intermediate levels of coherence. If the amount of contrast has a significant effect on the measured quantity extracted from simulated imagery, then Light Tunneling is an inadequate model. However, depending on certain application details, the domain of validity may actually extend to cases where $\ell_{coh}$ is longer than the criterion discussed above. To examine this issue a little more closely, let us consider the transverse coherence length of the irradiance speckle fluctuations in the image plane. Consider again the imaging concept sketched in Figure 2. If the illumination has a long temporal coherence, then, at a given point in the image plane, the irradiance is the magnitude-squared of the sum of a large number of random phasors, and therefore the point irradiance has the probability density function (PDF) characteristic of laser speckle. The speckle irradiance contrast (standard deviation / mean) is 1 if the illumination is highly monochromatic, but tails off from 1 to 0 as $\ell_{coh}$ decreases relative to $\sigma_h$. An important property for present purposes is the transverse spatial correlation length of the irradiance in the image plane. Consider just the diffraction-limited case (zero turbulence) for simplicity. Qualitatively, we can see that if we consider two image points separated by roughly one Airy width, then for practical purposes a different set of Airy disks contribute to the net field and irradiance at the two points. It follows that the transverse correlation length of irradiance must be roughly equal to the Airy disk width. This claim can be demonstrated more rigorously by using the detailed formulas of Fresnel propagation [(2)]. Now there are two factors that may reduce the importance of the speckle structure, thus allowing Light Tunneling to be a reasonable approach even when the illumination has substantial temporal coherence. First, depending on the application, the sensor pixel in the image plane may be significantly larger than the Airy disk size: in that case, we would obtain extra spatial averaging of the speckle contrast, and the results would tend further towards the incoherent limit. Second, it is possible that the image feature of interest is relatively insensitive to the presence of superposed fine-scale speckle. A possible example is the correlation tracking of an image feature like a corner: as long as the feature is defined over a span significantly larger than the speckle, the superposed fine-grained speckle may have a small effect on the key correlations. Obviously, the second mitigating factor that we list here is somewhat vague, and highly scenario-dependent.

There is another aspect of Light Tunneling where some conceptual difficulty exists: this is the question of consistency between the propagation model used to simulate the incident illumination, and the propagation calculation from reflector to imager plane. The Light Tunneling propagation model only deals

with the reflector-to-imager leg. The light incident on the reflector must be modeled by other existing WaveTrain methods. None of these methods is particularly well adapted to the completely incoherent limit, which is where the Light Tunneling method is on firmest ground. For example, suppose we wanted to model the imaging of a complex scene illuminated by spectrally-filtered solar and sky light. In this case, the incident irradiance would likely be highly uniform, so the best WaveTrain model would be to illuminate the reflector with a monochromatic *UniformWave*, at a wavelength near the band center, *without* propagating through any turbulence. When Light Tunneling is then used to propagate through the atmospheric turbulence from reflector to imager, we assume that a single wavelength is adequate to generate the PSF parameters (and lack of rough-surface speckle) associated with a physical bandwidth of, say, 50-100 nm.

### 3.3 Summary remarks on model validity

We have seen that the Light Tunneling method invokes two types of approximations, namely the restriction to low-order PSF moments with spatial interpolation, and the incoherent PSF sum. From the preceding discussion, we have seen that the validity of Light Tunneling may depend in rather complicated ways on details of the application. We have not yet tested the domain of validity, nor indeed the execution time improvement, in a detailed, quantitative way. Testing of the Light Tunneling procedure has been limited to qualitative inspection of perturbed imagery, and some qualitative comparison with field data. Visually, the judgment has been that reasonable agreement between simulated and field imagery can be obtained. Section 6 shows an example. More detailed testing is needed to establish the degree of validity of the Light Tunneling procedure, and to document the execution time improvement. The issue is complicated by the fact that "validity" can depend on the type of subsequent processing attempted with the simulated imagery, as much as on the propagation parameters per se. For example, an area of particular interest is the optical tracking of features in a complex scene. A relevant fact from a related but somewhat different problem is the following. From past experience with other numerical propagation methods for coherent light, we know that centroid tracking performance against the image of an extended, uniformly illuminated rough reflector can be nearly as good as tracking against a point beacon at the reflector center. The only difference is some extra high-frequency noise observed in the rough-reflector centroid signal, due to the fine-grained speckle structure in the image plane. We speculate on this basis that the Light Tunneling restriction to the low-order moments, coupled with the incoherent PSF sum may be a reasonable approach for such applications even in highly coherent illumination.

## 4 Possible modifications of Light Tunneling

Motivated in part by aspects of the validity discussion in Section 3, we briefly mention a few potential modifications or extensions of the Light Tunneling scheme. None of these features are present in the current WaveTrain implementation, but some are under current investigation. Each of the following paragraphs briefly discusses one possible modification.

It may be possible to create a different version of Light Tunneling that is better adapted to the perfectly coherent limit, but which would run almost as fast as the incoherent Light Tunneling. The only change this would require is to additionally assign a random overall phasor to each propagated and interpolated PSF (to represent the random surface height). The phasor would be completely uncorrelated from PSF to PSF, and would have a uniform phase distribution over (modulo) $2\pi$. This would be consistent with the physical fact that the surface height distribution is uncorrelated on the distance scale of the interpolated PSF grid. In contrast to the PSF moments, no interpolation would be required for this overall phase property, because of the complete lack of spatial correlation of the height phasors. Then, the net irradiance image would be computed by first adding the PSF complex fields in the image plane (NOTE: the only phase associated with each Gaussian-fitted PSF would be from the overall source-height phasor). As long as the same grids could be used, this modified procedure would require essentially no extra time compared with the present Light Tunneling. In order to produce the correct speckle contrast, some minimal number (perhaps 10 to 20?) of PSFs must substantially contribute at any image point: this may require a higher density for the interpolated-PSF grid than before, but that does not affect the physically-propagated PSF grid so the impact on execution time may be minimal.

In the current WaveTrain implementation, only a static reflector scene is allowed, although the turbulence disturbance can of course be dynamic (dragged phase screens). The static-reflector restriction has nothing to do with the Light Tunneling approximation concepts. It would be a simple matter to replace the static parameter that currently specifies the reflectance (BRDF) map with an input map that is allowed to be time-varying. This change would allow the user to model, for example, a time-varying pristine scene in which the track feature is momentarily obscured by some intervening object.

The current Light Tunneling procedure obtains the PSF moments by first physically propagating a sparse set of point sources, using numerical Fresnel propagation. It may be possible to obtain realizations of the moments without requiring any Fresnel propagations, by using further analytical results for propagation through turbulence to statistically characterize the moments. One approach is currently under investigation.

The current assumption of Gaussian PSFs fitted to propagation-derived low-order moments allows for different x-y asymmetries in each PSF, but the resulting PSFs still have some symmetries that are not present in the true PSFs. We conjecture that this is not a serious issue for the turbulence disturbance of extended scenes, but it may be more serious for disturbances that have a deterministic asymmetry component, such as thermal blooming. In the blooming case, all the disturbed PSFs have a common crescent-shaped asymmetry that may be poorly modeled by the Light Tunneling Gaussians. One should be able to use theoretical knowledge of this particular type of disturbance to define a more correct fitted PSF.

## 5 WaveTrain implementation and usage instructions

The WaveTrain implementation of Light Tunneling computes instantaneous images according to the discrete sum defined in Equation (4), given mesh data, reflectance map data, and incident irradiance. The

special modules created to implement Light Tunneling in WaveTrain are gathered into the two top-level modules *ExtendedPointSourceReflector* and *ExtendedImager*. Each of these is a composite WaveTrain system, i.e., each is composed of other WaveTrain library modules. For most Light Tunneling usage, users need only insert the two top-level systems into the user system, and connect and set those inputs, outputs and parameters. Users who wish to learn more about the internal workings of Light Tunneling, or who wish to create specialized variants of Light Tunneling, can use the WaveTrain System Editor to open the two top-level systems and study the structure and parameters at the lower hierarchy levels.

To construct a WaveTrain system that uses Light Tunneling, the user must:

(a) Propagate light to the reflectance scene using any of the usual WaveTrain methods and modules. As discussed in Sections 1 and 3, the new Light Tunneling modules are not used in any way in generating the incident light. This must be done in a consistent fashion using other existing WaveTrain modules.

(b) Represent the reflectance scene using the new WaveTrain module *ExtendedPointSourceReflector*. The key input parameter is a reflectance map that represents the BDRF of the scene.

(c) Propagate the light scattered by *ExtendedPointSourceReflector* through the desired atmosphere, using any of the usual WaveTrain modules such as *AtmoPath*. In order to carry out this propagation efficiently, it is particularly important that the user not overspecify the span of the propagation mesh: special considerations that apply to the propagation mesh for Light Tunneling are discussed in Section 5.2.

(d) Use the new WaveTrain module *ExtendedImager* as the detector for light scattered by *ExtendedPointSourceReflector*. Physically, *ExtendedImager* represents the same type of imaging detector as WaveTrain's basic *Camera* module. The difference is that *ExtendedImager* contains code that specifically works together with *ExtendedPointSourceReflector*. As in the case of *Camera*, any type of aperture and extra focus may be placed in front of *ExtendedImager*: a common WaveTrain usage is to insert a *Telescope* system for this purpose. (NOTE: *ExtendedImager* by itself, just like *Camera*, does not define the imager aperture size; the user unfamiliar with this procedure should consult the WaveTrain User Guide [(1)] section on the *Camera* module).

Figure 3 shows an example of a simple, complete WaveTrain system that uses Light Tunneling. In this example, we have illuminated the reflectance scene with a monochromatic, uniform plane wave that has *not* been propagated through turbulence. As discussed in more detail in Section 3, this would be a reasonable representation of solar illumination seen, for example, through a 50- or 100-nm wide spectral filter. This light illuminates the *ExtendedPointSourceReflector* module, which creates the initial grid of point sources that model the reflector. The light that exits from the reflector module is labeled the "transmitted" light, according to the WaveTrain convention for any exiting light. The *AtmoPath* module contains phase screens and performs physical (diffractive) propagation on the point-source fields set up by *ExtendedPointSourceReflector*. After *AtmoPath*, we have the aperture/refocusing system *Telescope*, followed by the special Light Tunneling receiver system *ExtendedImager*.

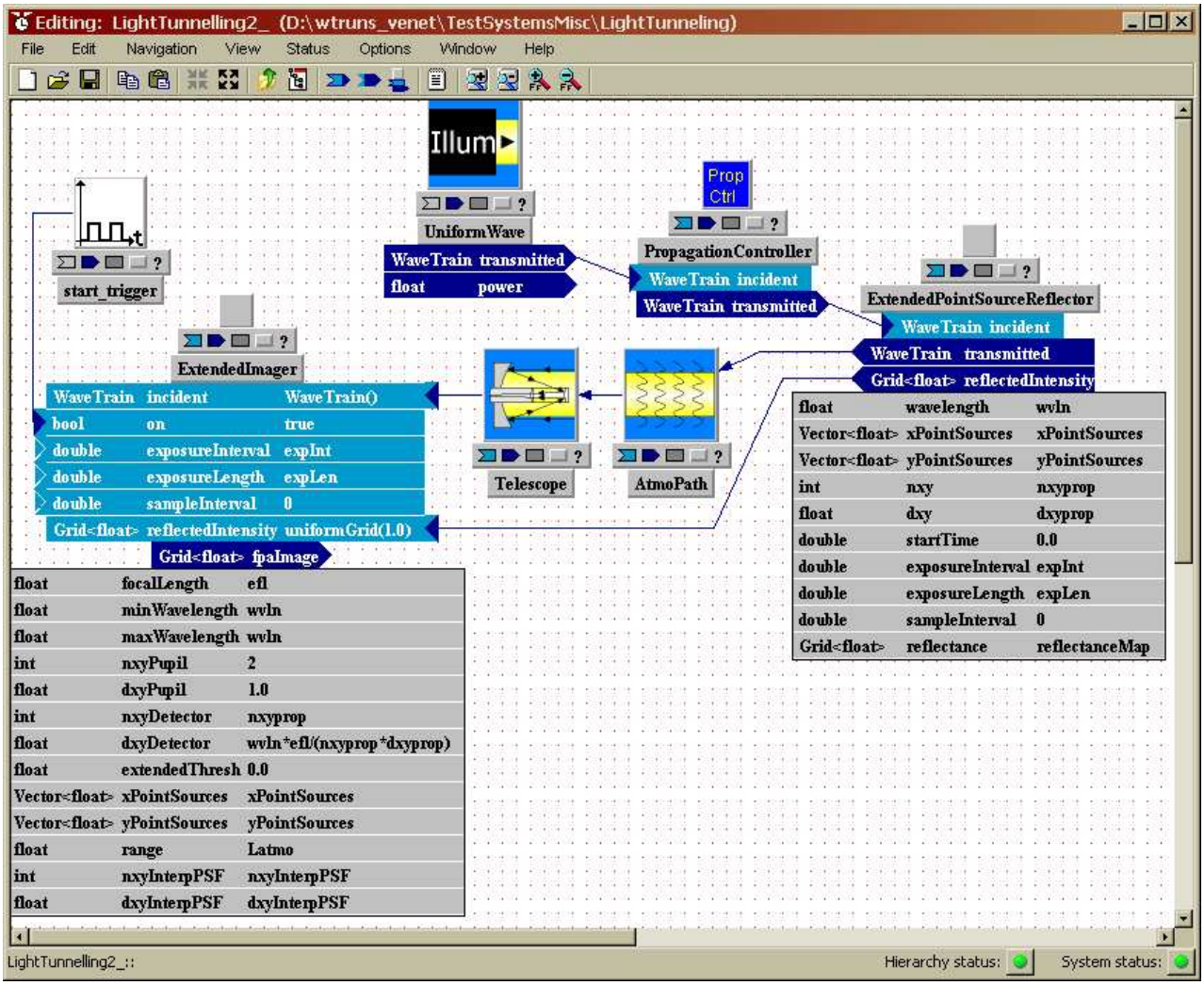


**Figure 3: A WaveTrain example system that uses Light Tunneling**

In the following sections, we explain usage and parameter details of the top-level Light Tunneling modules. There are numerous spatial sampling specifications in the module parameters, which are related to each other in rather complicated ways. In the following subsections, we discuss each parameter, input and output in turn, and in Section 6 we give a complete numerical example of key parameter settings and their interplay.

## 5.1 Parameters and I/O of *ExtendedPointSourceReflector*

### 5.1.1 Parameters of *ExtendedPointSourceReflector*

**`wavelength`:** incident wavelength (meters) to which the reflector should respond. Although the illumination must be modeled in terms of a single discrete wavelength, the physical illumination being modeled frequently has non-zero bandwidth (see the discussion in Sections 2 and 3). The wavelength will

affect the diffracted width of PSFs generated at the Light Tunneling receiver, and will affect the interaction of the propagating light with the phase screens in *AtmoPath*.

**`xPointSources`** and **`yPointSources`**: two vectors that specify the (x[k],y[k]) coordinates (in meters) of the k'th point source that will be *physically propagated* in the Light Tunneling algorithm. (Recall that other point-spread functions will be computed by interpolation from the physically propagated ones). Specification of the physically propagated point-source coordinates will dominate the execution speed of the propagation calculation.

NOTE: The physically-propagated point-source coordinates are *not* required to be on a uniform rectangular mesh: in some applications, it was thought reasonable to use a mesh that is more densely spaced in a region of high interest. Consequently, the x[k] and y[k] vectors must always have lengths equal to the *total number* of point sources, even if a uniform mesh of sources is desired. If a uniform mesh of (nx)×(ny) point sources is desired, spaced by dx and dy, there are two special WaveTrain/tempus library functions that can be used to define the setting expressions for x[k] and y[k]: these are, respectively,

`MeshXVecF(nx, ny, dx)` to create the x[k] vector

`MeshYVecF(nx, ny, dy)` to create the y[k] vector.

If a non-uniform mesh is desired, then a custom construction must be done: one way of accomplishing that would be to create the vectors in Matlab and read them in using the `mliLoad(…)` function (see the WaveTrain User Guide [(1)]).

**`nxy`** and **`dxy`**: linear point dimension and spacing (meters), respectively, of the square, uniform mesh on which the incident illumination is sampled. We recommend setting these parameters equal to the incident propagation mesh dimension and spacing. No particular registration is required between (`nxy`, `dxy`) and (`xPointSources` , `yPointSources`): WaveTrain will interpolate as needed.

The following four timing inputs may seem peculiar in the context of a reflector module: it may seem illogical for a reflector module to require timing specifications, and indeed WaveTrain's other reflector modules do not. However, the internal programming logic of Light Tunneling modules is somewhat outside the WaveTrain norm, and this accounts for the presence of timing parameters here (in addition to the *ExtendedImager*, where one expects timing parameters). The timing parameters have the same names and general significance as in any WaveTrain sensor (see the WaveTrain User Guide [(1)] section on Sensor Timing and Triggering for a general review).

**`startTime`**: simulation time (sec) at which the reflector begins acting. This should be set to account for the propagation delay relative to the *ExtendedImager*, in accordance with the usual WaveTrain delay patterns.

**`exposureInterval`**, **`exposureLength :`** the usual exposure window timing parameters in any WaveTrain sensor system (see the WaveTrain User Guide [(1)] section on Sensor Timing and Triggering for review of the details). In the present context, the typical setting would be to assign values equal to the

`exposureInterval` and `exposureLength` desired for the "physical" sensor, i.e., the *ExtendedImager*.

**`sampleInterval`**: typically, sampleInterval = 0. Recall (see the WaveTrain User Guide [1]) that non-zero values of `sampleInterval` are only used in WaveTrain to cause multiple propagations within one `exposureLength` window.

**`reflectance`**: the BRDF map of the scene, with respect to the transverse $\vec{\rho} = (x, y)$ coordinates. For input convenience, the sample points at which `reflectance` specifies the BRDF need not register with the propagation mesh, nor with the (`nxy`, `dxy`) or (`xPointSources` , `yPointSources`) meshes. WaveTrain will interpolate the data as needed. Note that `reflectance` must be supplied as a "Grid<float>" data type: see the WaveTrain User Guide [1] section on data types for further explanation. If the reflectance map is not normalized to BRDF units, the consequence is that the final Light Tunneling image will not be normalized to J/m$^2$; however the spatial distribution of irradiance will still be correct.

### 5.1.2 Inputs of *ExtendedPointSourceReflector*

**`incident`**: physically-propagated light that illuminates *ExtendedPointSourceReflector*.

### 5.1.3 Outputs of *ExtendedPointSourceReflector*

**`transmitted`**: light that exits *ExtendedPointSourceReflector* from the model point sources. Typically this light is input to an atmospheric propagation module, as in Figure 3.

**`reflectedIntensity`**: the product of the incident illumination map and the BRDF map, denoted $I_{inc}B$ in the previous theory sections. Figure 3 shows the required connection to pass this information directly to the *ExtendedImager* module. Notice that this connection *does not* go through an atmospheric path module.

## 5.2 Parameters and I/O of *AtmoPath*

*AtmoPath* is a general WaveTrain system, not specific to Light Tunneling. When used with Light Tunneling, *AtmoPath* serves to define the turbulence screens and to physically propagate the point sources specified on the (`xPointSources` , `yPointSources`) mesh. Other, analogous WaveTrain propagation modules could be used instead of *AtmoPath*. In order to carry out these propagations efficiently, it is particularly important that the user not overspecify the span of the propagation mesh. The key point to understand is that the mesh need only be large enough to allow for the physical propagation of a *single* point source from the reflector plane to the imager aperture. In particular, the propagation mesh need *not* span the entire reflector scene. It is vital to remember this, since the reflector scene is frequently much wider than the aperture for Light Tunneling applications. The "super-aperture" method should be used to propagate the point sources, so the super-aperture parameter in AtmoPath will be the key value that limits how small one can make the span of the propagation mesh.

In contrast to the propagation mesh, the phase screens, which are also specified in AtmoPath, must be large enough to span the entire reflector scene (and more, to accommodate the desired relative motion in the

simulation). Fortunately, large phase screens are usually not a dominant factor in determining simulation execution time.

The above considerations regarding required spans of the propagation mesh and the phase screens are consistent with the way WaveTrain meshes are handled for all off-axis WaveTrain sources and sensors (see the WaveTrain User Guide [1] for a general discussion).

Section 6 gives a numerical example of Light Tunneling mesh parameters to clarify the preceding statements.

## 5.3 Parameters and I/O of *Telescope*

*Telescope* is a general WaveTrain system, not specific to Light Tunneling. When used with Light Tunneling, *Telescope* has exactly the same functions as it would if followed by a *Camera* module (see the WaveTrain User Guide [1] section on the *Camera* module):

(a) *Telescope* applies an annular (possibly unobstructed) aperture to the incident beams.

(b) *Telescope* applies a focus adjustment (via its `range` parameter), so that the focal plane of *ExtendedImager* (or *Camera*) is the optical image plane exactly conjugate to the reflector scene. As always in WaveTrain, if an aperture shape other than annular is desired, then *Telescope* can be replaced by separate aperture and *Focus* modules.

## 5.4 Parameters and I/O of *ExtendedImager*

### 5.4.1 Parameters of *ExtendedImager*

**`focalLength`**, **`minWavelength`**, **`maxWavelength`**, **`nxyPupil`**, **`dxyPupil`**: these have functions identical to their functions in WaveTrain's *Camera* sensor system (see the WaveTrain User Guide [1] section on the *Camera* module). Recall in particular that `nxyPupil` and `dxyPupil` are *only* used if we explicitly invoke WaveTrain's specialized "wave-sharing" procedure. Normally, these two parameter values have *no* effect whatsoever in the WaveTrain simulation (although they must be assigned numerical values to allow compilation). In particular, note that these two parameters do *not* define the aperture size of the imager.

The presence of `minWavelength` and `maxWavelength` parameters may seem pointless here, since *ExtendedPointSourceReflector* only reflects at one discrete model wavelength. The reason for this min-max parametrization is that ultimately the ExtendedImager is based on WaveTrain's fundamental *Camera* module, which was set up to sense a range of wavelengths.

**`nxyDetector`**, **`dxyDetector`**: the linear point dimension and mesh spacing (meters), respectively, of the image-plane mesh on which each physically-propagated PSF is evaluated (*prior* to the moments calculation). We recommend that `dxyDetector` always be set equal to $(\lambda f)/(nxyprop \cdot dxyprop)$, where $f$ is `focalLength`, and $(nxyprop \cdot dxyprop)$ is the full width of the propagation mesh, in m, in the pupil plane of the receiver. This is the same dxy setting as the one recommended for WaveTrain's basic

*Camera* system: the motivation can be reviewed by reading the WaveTrain User Guide [1] discussion of the *Camera* system. With this spacing, the maximum useful value of `nxyDetector` is the *nxyprop.* setting of the *AtmoPath* module.

NOTE 1: in *ExtendedImager*, the product (`nxyDetector` × `dxyDetector`) only needs to span sufficient area to contain an *on-axis* PSF; i.e., this mesh need *not* span the whole array of displaced PSFs.

NOTE 2: (`nxyDetector`, `dxyDetector`) specifies the mesh on which the point-source PSFs are computed. *However*, the mesh on which the final extended image is reported is determined by (`nxyInterpPSF`, `dxyInterpPSF`) and the imaging magnification: see the discussion of those parameters below.

**`extendedThresh`**: relative threshold (0 to 1) for PSF irradiance, applied prior to computing the moments of the physically-propagated point-spread functions. (0 ⇒ use entire PSF).

**`xPointSources`**, **`yPointSources`**: the values assigned should be identical to those already assigned in *ExtendedPointSourceReflector*.

**`range`**: propagation distance (meters) from *ExtendedPointSourceReflector* to *ExtendedImager*.

**`nxyInterpPSF, dxyInterpPSF:`** linear point dimension and mesh spacing (meters), respectively, of the *reflector*-plane mesh on which the interpolating point sources are placed. That is, an interpolated PSF is generated for each of these intermediate source points: this is the mesh denoted by the indices $(l,m)$ in the theory sections, and it should span the reflector scene.

NOTE: the `dxyInterpPSF` spacing is specified in object space, i.e., the transverse space of the reflector. The corresponding spatial mesh coordinates in the imager sensor space can be determined by scaling with the imager magnification (*M* = -focalLength/range), but that is not what must be entered here. Still, *the principal significance* of (`nxyInterpPSF`, `dxyInterpPSF`) is that, after *M* scaling, they determine the mesh on which the final output image (`fpaImage` below) exists.

### 5.4.2 Inputs of *ExtendedImager*

**`incident`**: physically-propagated point-source light originating from the *ExtendedPointSourceReflector*.

**`on`**: the usual start-trigger input for any WaveTrain sensor system. As usual, this can be triggered with a square wave module to specify an arbitrary start time for the sensor.

**`exposureInterval`**, **`exposureLength:`** the usual exposure window timing parameters in any WaveTrain sensor system (see the WaveTrain User Guide [1] section on Sensor Timing and Triggering for review of the details).

**`sampleInterval`**: typically, sampleInterval = 0. Recall (see the WaveTrain User Guide [1]) that non-zero values of `sampleInterval` are only used in WaveTrain to cause multiple propagations within one `exposureLength` window.

**`reflectedIntensity`**: the $I_{inc}B$ product map, obtained by connecting to the corresponding output of *ExtendedPointSourceReflector*, as shown in Figure 3.

### 5.4.3 Outputs of *ExtendedImager*

**`fpaImage`:** the final image of the *ExtendedPointSourceReflector* scene, as computed by Light Tunneling. If the reflectance map was specified in BRDF units, then the image units will be J/m$^2$ in the image plane of *ExtendedImager*. (The units and radiometry conventions are consistent with the other principal WaveTrain reflector modules and sensors. Note that the theory background was phrased in terms of W/m$^2$, but the WaveTrain implementation follows the principle that all sensors are temporally-integrating sensors.)

The mesh coordinates on which `fpaImage` exists are determined by scaling (`nxyInterpPSF`, `dxyInterpPSF`) by the imager magnification (*M* = -focalLength/range): see the above discussion of parameters (`nxyInterpPSF`, `dxyInterpPSF`).

## 5.5 Miscellaneous usage and implementation comments

By splitting and combining wavetrains, Light Tunneling can be combined with other types of propagation methods in the same WaveTrain system. The principal constraint is that *ExtendedPointSourceReflector* and *ExtendedImager* must usually be used as a pair. Other WaveTrain sensors are not designed to work with *ExtendedPointSourceReflector*, and other WaveTrain sources are not designed to work with *ExtendedImager*. Section 5.5.2 gives some further details on reflector-sensor pairing options.

### 5.5.1 Diagnostics

A useful diagnostic for investigating the performance of Light Tunneling would be the recording and inspection of the physically propagated PSFs. At present, the user interface to the *ExtendedImager* does not have the option to record individual PSFs. However, what can be recorded are 2-D maps of all the moments of the interpolated PSFs that arise in the Light Tunneling procedure. This allows the user to verify, for example, that the tilts and standard deviations of the PSFs are behaving in a plausible manner with plausible magnitudes. From inspection of Figure 3, we see that the PSF moments are not provided as outputs of *ExtendedImager*. However, the moments *are* still available for recording in the "Recorded Outputs" dialog box of the WaveTrain Run Set Editor (because *ExtendedImager* is a "composite" system). In the list of available outputs under subsystems of *ExtendedImager*, the user will see the 2D arrays "interpIntensity", "interpRho", "interpXblur", "interpXtilt", "interpYBlur", and "interpYTilt". "interp{X,Y}Blur" are the standard deviations of all the PSFs used to construct the final image, in units of meters in the object (reflector) space. Similarly, "interp{X,Y}Tilt" are the displacements of the PSF centroids from their unaberrated positions, again in units of meters in the object (reflector) space. Image plots of these 2D arrays with a color-bar will show the magnitudes and distribution of the standard deviations and centroid displacements over the reflector scene.

### 5.5.2 Composition of *ExtendedPointSourceReflector* and *ExtendedImager*

In the WaveTrain System Editor, we can view the internal composition of *ExtendedPointSourceReflector*. The subsystem *ExtendedPointSource* is the key new component of the composite reflector system. Likewise, by viewing the internal composition of *ExtendedImager*, we see that the subsystem *ExtendedCamera* is the key sensor component of the imager system. For customized work or testing purposes, users can construct new systems directly from *ExtendedPointSource* and *ExtendedCamera*, which are separately available in the WaveTrain component library: the *ExtendedPointSource* - *ExtendedCamera* pair is designed to work together, but generally *ExtendedPointSource* cannot feed other WaveTrain sensors, nor can *ExtendedCamera* handle light from other WaveTrain sources.

### 5.5.3 Interpolation algorithm

If the physically-propagated point sources lie on a non-uniform mesh, then the interpolation of the PSF moments is done using a Fortran library subroutine, *IDBVIP()* , which uses triangulation and bivariate quintic polynomial fitting. However, if the code detects a uniform mesh, then a faster interpolation algorithm is used. The choice of interpolation algorithm is significant, because initial testing suggests that a substantial fraction of the Light Tunneling execution time can be taken up by the interpolation calculations.

### 5.5.4 Parallel processing

For users who can run WaveTrain parallelized code on a multi-processor system, Light Tunneling execution time benefits greatly from parallel processing. The algorithm parallelizes simply, since each point-source propagation is entirely independent: the parallel implementation assigns the propagation of the different point sources to different processors.

## 6 Example parameter settings and imagery

The numerous spatial sampling specifications in the module parameters listed above may seem confusing. In the present section, we give a numerical example of key parameter settings; we hope this will clarify some aspects of the interplay between the specifications.

### 6.1 Example system – numerical values of key parameters

Consider the viewing of an extended scattering scene along a roughly horizontal line of sight, with illumination having broad enough bandwidth to eliminate rough-surface speckle. Suppose we have the following scenario parameters:

- Nominal imaging wavelength = 0.6 $\mu$m
- Propagation distance $L$ = 5 km
- 8 equal-strength phase screens, uniformly distributed along the path
- Uniform $C_n^2$ strength profile, values given in Table 2.

- Reflecting scene 2 x 1.95 m, given on mesh spacing = 0.5 cm
- Imager aperture diameter = 30 cm

For general orientation, we derive integrated turbulence strength parameters for the above scenario from the usual analytic approximations ($r_0$ is the Fried coherence length, and $\theta_0$ is the isoplanatic angle):

| $C_n^2$ ($m^{-2/3}$) | $r_0$ (cm) (sph wav) | $\frac{\lambda}{r_0} L$ (cm) | $L\theta_0$ (cm) |
|---|---|---|---|
| 1E-16 | 27 | 1.1 | 8.6 |
| 1E-15 | 6.9 | 4.4 | 2.2 |
| 5E-15 | 2.6 | 11 | 0.82 |

**Table 2: Integrated turbulence parameters for Light Tunneling example**

With the tabulated parameters in mind, we specify the following key propagation, phase screen and imaging parameters in the WaveTrain Light Tunneling components:

**Illumination incident on reflector scene:** we illuminate the reflectance scene with a monochromatic, uniform plane wave that has *not* been propagated through turbulence. We use the uniform intensity wave to represent the essentially unscintillated uniform illumination that would be obtained from ambient solar light. We assume that a single wavelength is adequate in the present model to represent the PSF widths (and lack of rough-surface speckle) obtained with a physical bandwidth of, say, 50-100 nm.

**`reflectance`** (in *ExtendedPointSourceReflector*)**:** the scene of dimensions (Wx, Wy) = (2.00, 1.95) m, on a mesh spacing of (0.5, 0.5) cm

**`nxy, dxy`** (in *ExtendedPointSourceReflector*)**:** mesh on which the incident illumination is sampled: let dxy = 0.5 cm, nxy = 400. (Note that if **`dxy`** were different than the reflectance map mesh, WaveTrain would interpolate as needed.)

**`xPointSources`** and **`yPointSources`** (in *ExtendedPointSourceReflector*)**:** the physically-propagated point sources: let us use a uniform mesh of 20 x 20 point sources, with uniform spacing 9.75 cm.

**Propagation mesh in *AtmoPath* (from reflector to imager):** let use a mesh with equal spacing at reflector and receiver, and let us choose Fresnel spacing: i.e., we obey the constraints $nxyprop = \lambda L / dxyprop^2 = (nxyprop \cdot dxyprop)^2 / \lambda L$. Since the imager aperture is 30 cm, let us choose a mesh width $(nxyprop \cdot dxyprop)$ of at least 60 cm. From the Fresnel constraint, this requires at least *nxyprop* = 120, so we choose 128 as the next power of 2. The Fresnel constraint then requires *dxyprop* = 0.484 cm. The resulting mesh width $(nxyprop \cdot dxyprop)$ = 61.95 cm.

NOTE: the propagation mesh width does NOT span the 2-m reflectance map, BUT it is sufficient to accurately represent any single point-source propagation, as discussed in Section 5.4.1.

**`focalLength`** (in *ExtendedImager*)**:** suppose the effective focal length of the imager = 1.5 m.

**`nxyDetector`**, **`dxyDetector`** (in *ExtendedImager*)**:** recall this is the sensor-plane mesh on which the physically-propagated point-source PSFs are initially computed. As discussed in Section 5.4.1, we set these parameters to `dxyDetector` = $(\lambda f)/(nxyprop \cdot dxyprop)$ = 1.45 $\mu$m, and `nxyDetector` = *nxyprop* = 128

NOTE: the smaller $r_0$ from Table 2 is 6.9 cm, and $(\lambda/r_0)\cdot f$ = 13 $\mu$m. Therefore, the sensor-plane span (128*1.45$\mu$m) = 186 $\mu$m is decidedly sufficient to capture the PSF.

**`nxyInterpPSF, dxyInterpPSF`** (in *ExtendedImager*)**:** recall that this specifies the mesh on which the extended image is finally generated, but that these specs are to be given in *object*-space coordinates, i.e., in the plane of the reflector map. Let us specify `dxyInterpPSF` = 0.5 cm, `nxyInterpPSF` = 390 (this spans 1.95×1.95 m, which is essentially all of the input scene).

## 6.2 Example system – sample output

The upper left panel of Figure 4 shows a pristine scene used as reflectance map. The other panels each show one instantaneous realization of the scene image perturbed by turbulence, for each of the $C_n^2$ (equivalently, $r_0$) strengths, as computed by Light Tunneling with the parameters in Section 6.1. The Light Tunneling *ExtendedImager* module actually produces an inverted image, consistent with WaveTrain's *Camera* module, but we have erected the image for easier human consumption.

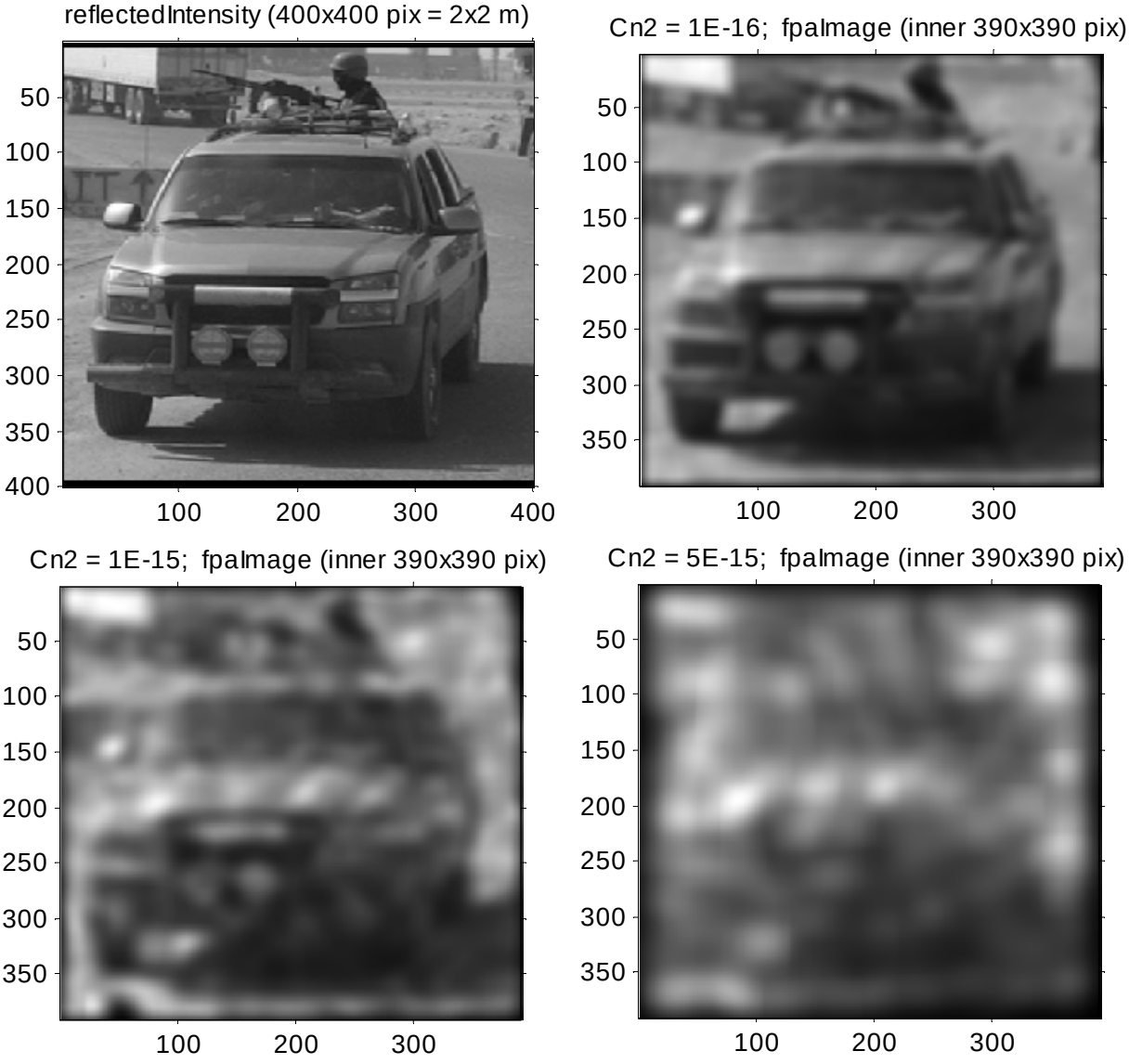


**Figure 4: Light Tunneling sample images. Upper left: Pristine image.**
**Upper right: Sensor image for Cn2 = $10^{-16}$ $m^{-2/3}$ ($r_0$ = 27 cm), rotated to match source orientation.**
**Lower left: Sensor image for Cn2 = $10^{-15}$ $m^{-2/3}$ ($r_0$ = 6.9 cm) , rotated to match source orientation.**
**Lower right: Sensor image for Cn2 = $5 \cdot 10^{-15}$ $m^{-2/3}$ ($r_0$ = 2.6 cm) , rotated to match source orientation.**

# 7 Radiometry details of the WaveTrain implementation

## 7.1 The normalization factor *N* in the Light Tunneling Image Equation

The usage instructions and example in Sections 5 and 6 provide the WaveTrain user all the information that is necessary to generate Light Tunneling propagation results in well-defined physical units. It is not necessary for the user to digest the contents of the present section. The present section is provided mainly as documentation for WaveTrain developers, but also for users who desire more information about the theoretical background of the Light Tunneling formulas. The main purpose of the present section is obtain the radiometric normalization factor *N* that appears in the final Light Tunneling Image Equation (4).

To obtain *N*, we compare Equation (4) with a corresponding result for the unaberrated image, where all the factors are known from basic radiative transfer theory. To make this comparison, we need to integrate out the PSF, which plays no role in the unaberrated image. To do this, we compute the long-time average of Equation (4) and then integrate over the entire image plane:

$$\begin{aligned} \overline{P_{img}} &= \int_{\infty} d^2\rho' \, \overline{I_{img}(\vec{\rho}'_{l',m'})} \\ &\approx \frac{1}{N} \sum_{l,m} \left[ \int_{\infty} d^2\rho' \, \overline{S^{gauss}(\vec{\rho}'_{l',m'}; \vec{\rho}_{l,m})} \right] \left[ I_{inc}(\vec{\rho}_{l,m}) \, B(\vec{\rho}_{l,m}) \, (\Delta^2 \rho_{l,m}) \right] \end{aligned} \tag{5}$$

where the overbar denotes long-time average. We also suppose that $I_{inc}$ is constant in time (we can consider any special case that makes it convenient to solve for the single unknown scalar *N*). Now the key observation is that the integral of the *time average* PSF in Equation (5) is independent of source index $(l,m)$, so that the integral can be moved outside the summation. Based on the definition of *S*, this integral is the time-average total power in the image plane due to a 1W/sr point source, or equivalently the time-average total power intercepted by the receiver aperture from such a source. Therefore we can write

$$\begin{aligned} \overline{P_{img}} &\approx \frac{1}{N} \overline{P_1} \sum_{l,m} \left[ I_{inc}(\vec{\rho}_{l,m}) \, B(\vec{\rho}_{l,m}) \, (\Delta^2 \rho_{l,m}) \right] \\ &= \frac{1}{N} \left( 1 \frac{W}{sr} \cdot \frac{a_{pup}}{z^2} \right) \sum_{l,m} \left[ I_{inc}(\vec{\rho}_{l,m}) \, B(\vec{\rho}_{l,m}) \, (\Delta^2 \rho_{l,m}) \right] \end{aligned} \tag{6}$$

where $a_{pup}$ is the area of the receiver aperture, and *z* is the receiver-to-reflector distance.

The last line of Equation (6) is ready to be compared with radiative transfer formulas that govern the unaberrated image. A formula for $P_{img}^{ua}$, the unaberrated (*ua*) total scattered power captured by the imager aperture, can be obtained using the concepts sketched in Figure 5. Starting from the definitions of radiance

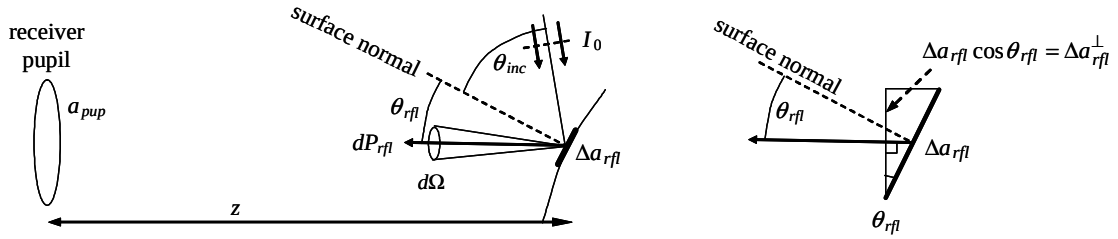


**Figure 5: Incoherent radiative transfer relevant to Light Tunneling normalization**

and BRDF, we consider the power scattered from the projected area element $\Delta a_{rfl}^{\perp}$ into the solid angle subtended by the imager pupil. That power is

$$\Delta P_{img}^{ua} = L_{rfl} \frac{a_{pup}}{z^2} \Delta a_{rfl}^{\perp} = I_{inc} B \frac{a_{pup}}{z^2} \Delta^2 \rho \tag{7}$$

The intermediate quantity $L_{rfl}$ is the reflected radiance (units: W/m$^2$/sr), $a_{pup}$ is the area of the receiver aperture, *z* is the receiver-to-reflector distance, and $I_{inc} = I_0 \cos\theta_{inc}$ is the power per unit area in the plane of the surface. Equation (7) assumes that the receiver pupil is nominally perpendicular to the line of sight, but the reflector surface may be generally tilted and curved. Summing over all the area elements of the surface yields the total power:

$$P_{img}^{ua} = \sum_{l,m} \Delta P_{l,m}^{ua} = \frac{a_{pup}}{z^2} \sum_{l,m} I_{inc}(\vec{\rho}_{l,m}) B(\vec{\rho}_{l,m}) \Delta^2 \rho_{l,m} \tag{8}$$

This result for $P_{img}^{ua}$ must equal the earlier expression for $\overline{P_{img}}$ given by Equation (6), which yields the following very simple result for the normalizing factor *N*:

$$\boxed{N = 1 \frac{W}{sr}} \tag{9}$$

This completes the definition of the Light Tunneling Image Equation (4).

## 7.2 Irradiance versus energy density in the WaveTrain implementation

The Light Tunneling Imaging Equation (4), and the radiometry details were developed in terms of irradiance (units: W/m$^2$). This is the usual quantity to use for theoretical development. A slight complication in the WaveTrain implementation is that all WaveTrain sensors actually output the energy density J/m$^2$, which is the irradiance, W/m$^2$, times the `exposureLength` parameter of the sensor. To make Light Tunneling consistent with this principle, WaveTrain actually evaluates the J/m$^2$ equivalent of Equation (4). The intermediate PSF quantities, *S*, are evaluated by WaveTrain in J/m$^2$, and then the final image in the sensor plane of *ExtendedImager* , `fpaImage` (see Section 5.4.3), also emerges in J/m$^2$.

## 8 References

1. www.mza.com/wavetrain: WaveTrain documentation index, WaveTrain User's Guide.
2. Goodman, J.W., "Statistical Properties of Laser Speckle Patterns", in *Laser Speckle and Related Phenomena*, 2nd ed., pp. 35 ff., J.C. Dainty, ed., Springer-Verlag, 1984.