Algorithms for prediction of indoor radio propagation

Steven Fortune
Bell Laboratories
Murray Hill, NJ 07974
sjf@research.att.com

January 7, 1998

Abstract

The design of indoor wireless communication networks requires prediction of radio propagation inside a building. Radio propagation inside a building is complex, because each building wall can act both as an obstruction, attenuating a propagation path, and as a mirror, providing an additional reflecting path. We describe two algorithms that simulate radio propagation, a ray tracing algorithm that uses a discrete sample of propagation directions and a beam tracing algorithm that enumerates reflection cones defined by building walls. The two algorithms are compared experimentally and analytically. With a triangulation-based spatial data structure, both algorithms are fast enough to provide propagation simulations in a few minutes of computing time, even for large buildings.

1 Introduction

Wireless communication networks that use radio frequency transceivers are important for many indoor applications. A typical network has a few fixed base transceivers, which communicate with a larger outside network, and possibly many portable transceivers, which move around the environment. One such network might extend outdoor cellular telephone service to the inside of a large building, avoiding "air time" charges for indoor calls. Another network might allow portable point-of-sale terminals in a department store to communicate with a base transceiver attached to a bookkeeping computer.

A common design problem of such systems is the placement of base transceivers to ensure adequate signal coverage throughout the environment. Radio propagation is sufficiently complicated and sufficiently dependent upon building construction that site-specific design is required. The communication hardware is sufficiently expensive that it is unacceptable to over-engineer a design. A desirable design tool would predict radio frequency propagation, given a description of the building, the location of a base transceiver, frequency, transmitted power, and so forth. The tool could be applied as needed, to try alternative design scenarios or to optimize base placement.

A simple model of radio propagation, based on geometric optics, provides the basis for such a design tool. A building is a list of walls, each a polygon labeled with construction material. Radio from a point source propagates along a straight line, with power diminishing according to an inverse-square law. At a wall, part of the radio power passes through and part is reflected specularly, with proportions depending upon material and incidence angle. Diffraction is ignored in this model, as well as scattering at a wall or because of furniture and other clutter. The unpredictable position of portable transceivers can be addressed by producing a coverage map, that is, the received power at a set of sample points throughout the building. Typically the sample points are chosen as the lattice points of a regular grid. This radio model is described in more detail in section 2 below.

Naive simulation of radio propagation is computationally challenging. In a large building there may be thousands of possible propagation paths between base and sample point, though only a few contribute significant power, and received power may need to be simulated at thousands of different sample points. The dynamic range of the simulation—the ratio between minimum and maximum received power—is 10^6 or more. To make the simulation as useful as possible, it should be accurate over the entire dynamic range.

This paper describes three algorithms that can be used to predict radio propagation. The first, the shadowing algorithm described in section 3, is a direct simulation of the wavefronts produced by wall reflections. Each wall can be viewed as a partially reflecting mirror that illuminates a polyhedral reflection cone using reflected power from the base. For each reflection cone, the shadowing algorithm maintains the partition of the wavefront induced by wall shadows, i.e. by the transmission loss through intervening walls. Though this algorithm is conceptually attractive, it appears to be too expensive computationally to be practical.

The second algorithm, the beam tracing algorithm described in section 4, can be viewed a practical variant of the shadowing algorithm or as variant of typical beam tracing algorithms [15]. The beam tracing algorithm enumerates reflection cones without maintaining the wall shadow partition. If a sample point lies in a reflection cone, then there is a propagation path to the sample point that reflects off the corresponding wall; the propagation path must be traced to determine transmission loss through intervening walls. All propagation paths up to some fixed number of reflections can be found using reflection cones in this manner. A significant optimization for the beam tracing algorithm is filtering, which eliminates reflection cones and individual propagation paths that cannot contribute significant power to a sample point.

The final algorithm is a forward ray tracing algorithm [15], described in section 5. A finite sample of the propagation directions from the base is chosen and a ray is launched for each direction. At each intersection with an obstacle, the ray splits into transmitted and reflected subrays. When a ray passes near a sample point, the current power of the ray is added to the accumulated power of the sample point. The ray tracing algorithm is optimized using ray splitting [4].

Section 6 describes a spatial data structure, based on triangulations, that can be used to represent the environment. It allows ray-shooting queries (report the first wall intersected by a query ray) to be answered in constant time, at least as measured experimentally; it also allows polyhedral cone queries (report the walls intersected by a polyhedral cone) to be answered efficiently. The triangulation data structure requires that all walls be horizontal or vertical (parallel to the z-axis); this restriction is not onerous for most buildings.

With the spatial data structure, both ray tracing and beam tracing are plausible prediction algorithms. It is not obvious which algorithm is better computationally. Both algorithms have a large number of parameters that allow the completeness of a simulation to be traded off against running time. Sections 4 and 5 give analytic worst-case estimates of the running time of both algorithms, but the estimates are pessimistic in practice and in fact are expressed in different problem parameters, making direct comparison difficult.

Section 7 gives a comparison of the experimental run-times of the ray tracing and beam tracing algorithms. The comparison is reasonably fair, since both algorithms use the same spatial data structure. Neither algorithm dominates the other in the sense that each algorithm is fastest for some buildings and parameter values. In general, the beam tracing algorithm is better for a few reflections while the ray tracing algorithm is better for many reflections; the ray tracing algorithm is better if allowed path loss is small while the beam tracing algorithm is better if allowed path loss is large. Both algorithms suffer from sampling artifacts, where the power at a sample point is computed incorrectly. Sampling artifacts are much more severe for the ray tracing algorithm, where systematic errors are possible.

The ray tracing and beam tracing algorithms have been incorporated into a wireless system engineering tool called WISE [3]. WISE includes a powerful graphical user interface that allows system engineers to choose base placements interactively, visualize propagation predictions, and optimize base placement automatically. Both algorithms are fast enough that useful predictions can be obtained in a few minutes on a workstation, even for large buildings (200 meters by 200 meters, 1500 walls).

This paper does not address the accuracy of the propagation model itself. Many factors might affect prediction accuracy: the presence of furniture, inaccuracies in the building model, the modeling of reflection and transmission losses, other radio effects such as scattering or diffraction, etc. The literature contains some comparisons between predictions and measurements, for this and similar models [5, 14].

Other work. The 'image' method finds a propagation path by using the reflected image of a transmitter in the plane of a wall, iterated as necessary to find multireflection propagation paths. The image method has been used extensively for radio propagation [6, 14], as well as sound propagation [1]. Beam tracing can be viewed as a generalization of the image method; it has been independently proposed by Takahashi *et al* [13].

Ray tracing of course is similar to methods used in image rendering [15]. It too has been used frequently for radio propagation [4, 5, 9, 11, 12]. Kreuzgruber *et al* [4] suggest ray splitting (see section 5.2) to reduce computation time. Rajkumar *et al* [9] use ray tracing with a different spatial data structure, a binary space partition tree, to represent the environment.

2 Radio background

The books by Parsons [7] and by Ramo *et al* [10] are general references for radio propagation.

2.1 Propagation paths

The environment consists of a set of *walls*, each of which is a polygon with a small number of edges, e.g., a rectangle. Two walls should intersect only along an edge common to both. Each wall has a type that indicates its construction material and hence radio properties. A typical environment might have between 100 and 10000 walls. Figure 1 shows the ground floor of the AT&T Crawford Hill building; this model has about 80 walls. Figure 2 shows a single floor of the AT&T Mt. Airy building; the two floors together have about 2300 walls.

A *propagation path* is a directed polygonal chain such that each internal vertex lies on a wall and the

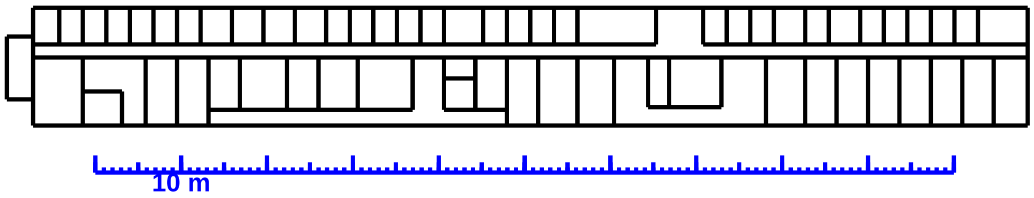


Figure 1: AT&T Crawford Hill building.

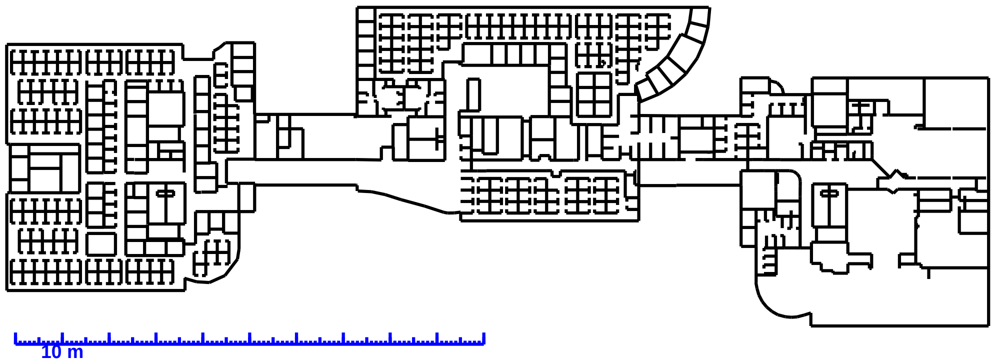


Figure 2: First floor of two-story AT&T Mt. Airy building.

path near each internal vertex is generated by reflection off the wall using Snell's law, that is, the direction along the path after the vertex is the reflection in the plane of the wall of the direction before the vertex. A propagation path may pass through a wall.

2.2 Path loss

A propagation path p (from base to sample point) has an associated *path loss*

$$l_p = \frac{\prod_i \rho_i \prod_j \tau_j}{|p|^2}. \tag{1}$$

Here $|p|$ is the length of the path in meters. The index i varies over reflections and ρ_i is the ith *reflection coefficient*, i.e., loss due to reflection. The index j varies over wall intersections that are not reflections and and τ_j is the jth *transmission coefficient*, i.e., loss due to transmission through a wall. The *path loss* to a sample point is given by

$$l = \sum_p l_p \tag{2}$$

where the sum runs over all propagation paths p from base to sample point. The path loss measured in dB is $10 \log_{10} l$.

2.3 Power

The (free-space) received power at a distance of one meter from the base is given by

$$\beta_{1m} = P \left(\frac{\lambda}{4\pi} \right)^2$$

where P is the power input to the transmitter and λ is the wavelength in meters. The *power* of path p is

$$\beta_{1m} l_p,$$

where l_p is the path loss of path p (equation (1)), and the *received power* at a sample point is

$$\beta = \beta_{1m} l \tag{3}$$

where l is the path loss to the sample point (equation (2)). Received power measured in dBm is $10 \log_{10} \beta$, assuming transmitter power is measured in milliwatts (mw).

A typical transmitting antenna used for a communication network might have an input power of 34 mw; at 2 Ghz this yields a free-space received power of about -23 dBm at one meter. A typical receiving antenna might require a received power of -70 dBm for reliable communication, implying a free-space coverage radius of about 220 meters. It is appropriate to model paths with even lower power, say down to -90 dBm, since several paths could add together to meet the -70 dBm threshold, and since it is desirable to model interference between transmitters operating on the same frequency.

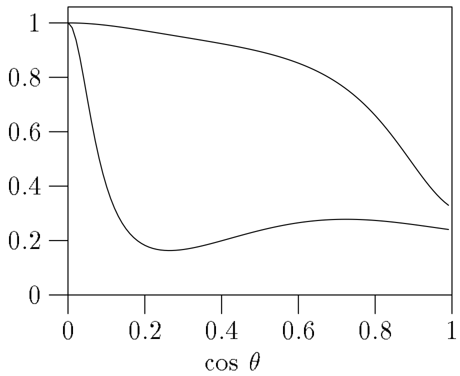


Figure 3: Theoretical reflection coefficient as a function of incident angle: sheetrock (top) and wood over concrete (bottom). Transmission coefficient is 1 minus reflection coefficient.

2.4 Coverage calculation

A *coverage calculation* determines received power at a set of sample points. Typically there are infinitely many propagation paths to a sample point. Hence a coverage calculation is usually an approximation, with propagation paths restricted to those of relatively high power or with a limited number of reflections.

2.5 Transmission/reflection coefficients

Theoretical transmission and reflection coefficients can be determined using a multilayer dielectric model [10] derived from estimated properties of construction materials. For example, a sheetrock wall might be modeled as 0.5 inches of plaster, 3.5 inches of air, and 0.5 inches of plaster. For a fixed wall material and wall orientation, the coefficients depend upon the angle of incidence of path to wall and can be determined to adequate accuracy with table lookup indexed by discretized angle. Figure 3 plots the coefficients for sheetrock and 'wood over concrete' as a function of incident angle; both walls are vertical. These materials are *lossless*, i.e., reflection and transmission coefficients sum to 1 at each angle; for a *lossy* material, the sum may be less than 1. For most walls, the transmission coefficient is maximized if the path is perpendicular to the wall; the reflection coefficient approaches 1 if the path is almost parallel to the wall. In practice, theoretical reflection and transmission coefficients may need to be altered to model losses due to scattering, construction inhomogeneities, furniture, etc.

2.6 Extensions

The formulation above implicitly assumes that the base and portable antennas are isotropic, that is, radiate power equally in directions. Directional antennas can easily be accommodated by additional gain determined by the initial and final directions of a path.

The wave nature of radio implies that each incoming path has an associated phase change (relative to the base antenna). The power received at a sample point depends upon the phases of the incoming paths. The received power can fade (i.e., fluctuate) dramatically in a small spatial neighborhood of the sample point, as the phases of the incoming paths change with respect to each other. The simple power summation in equations (2) and (3) does not model phase; however, it can be justified as the "local mean power", i.e., the received power averaged over a small neighborhood of the sample point. An alternative is to associate a phase with each path; the power at a sample point can then be computed by a summation over the complex numbers. The precise prediction of local fades would require extremely precise knowledge of base antenna and wall position. Phase prediction can be used to give a qualitative estimate of local fading, even with somewhat imprecise position information.

Each propagation path has an associated time delay, the length of the path divided by the speed of light. High-bandwidth communication may require that each path have approximately the same delay. A simple extension is to compute the *delay spread*, that is, the standard deviation of path delays weighted by path power.

3 A propagation model

For a fixed position of the base antenna, imagine transmit power being turned on for a very short interval. Radio propagates from the antenna at the speed of light. As long as propagation is in free space, the wavefront forms a sphere; at any given time the power on the surface of the sphere is invariant over the sphere.

At a wall w, part of the wavefront power is reflected and part is transmitted. The wavefront splits into two pieces. One wavefront is still a sphere as before, except that the power in the shadow of the wall is reduced by the transmission coefficient of w.

The second wavefront is formed by reflection off w. It sweeps out a *reflection cone* C, consisting of all points reachable by a single-reflection path off wall w. Let v be the reflected image of the base in the plane of the wall; v is a *virtual transmitter*. It is immediate that a sample point is reachable by a single reflection off wall w exactly if the line segment from v to the sample point intersects w at the reflecting point. (Figure 4). Thus the reflection cone is a truncated polyhedral cone. It has apex v; each plane through v and one of the edges of w bounds the cone; the cone is truncated by the plane of w, deleting the portion of the cone from v to w. See Figure 4.

The reflected wavefront may intersect a second wall w', spawning a second reflected wavefront that sweeps out a second reflection cone C'. The *(iterated) virtual transmitter* v' is the reflection of v in the plane of w'. As before, there is a propagation path to the sample point reflecting off w and w' in order exactly if the line segment from v' to the sample point intersects w' at the reflecting point of w'. The reflection cone C' has apex v'; the planes bounding C' are formed from v' and the edges of the polygon $w' \cap C$, as before.

Clearly this process continues indefinitely. Each intersection of a wavefront with a wall shadows the wavefront and spawns a new reflected wavefront sweeping out a new reflection cone.

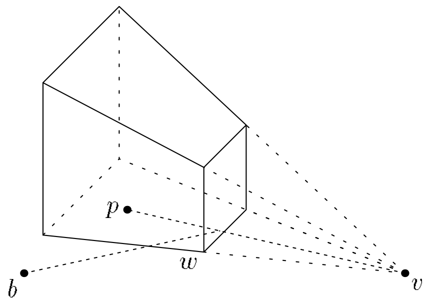


Figure 4: Reflection cone (outlined in solid lines). b is base, v is virtual transmitter. Reflecting path from b to p intersects w at same point as line segment vp.

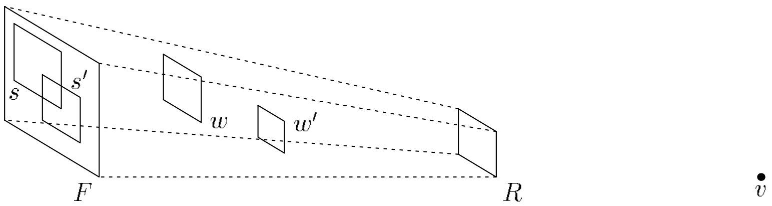


Figure 5: Shadow partition: walls w and w' have shadows s and s' on wavefront F; wall R is reflecting wall and v is virtual transmitter.

3.1 The shadowing algorithm

The shadowing algorithm performs a coverage calculation by simulating the expansion of wavefronts. The algorithm can be kept finite by considering wavefronts that arise from at most a fixed number of reflections.

At any given time, the wavefront of a fixed reflection cone is the intersection of the cone with a sphere centered at the virtual transmitter. The *shadow* function assigns to each point of the wavefront the sequence of walls intersected on the propagation path from base to the point. The algorithm maintains the *shadow partition* of each wavefront, i.e., the partition induced by the shadow function. This partition can be represented as a planar graph embedded on the wavefront; see Figure 5.

A simulated wavefront may pass through a sample point. The region of the shadow partition containing the sample point determines the sequence of walls intersected on the propagation path to the sample point, and hence the additional power contributed by the propagation path. (Note that power within a region may not be constant because of the angle dependence of reflection and transmission coefficients.) A wavefront may also pass through a wall, in which case the shadow partition is updated and a new wavefront spawned.

The shadowing algorithm would be rather complicated to implement, and in any event one of the other two algorithms is probably better in practice. Hence we do not give a detailed run-time analysis. However, we observe two properties. First, the number c of reflection cones in the worst case grows as n^r, where n

is the number of walls and r is the number of reflections. More typically, each reflection cone intersects $O(n^{1/2})$ walls, so c grows as $n^{(r+1)/2}$. Section 7.3 gives experimental values for c. Second, the maintenance of the shadow partition is expensive. Define the *propagation volume* of the reflection cone as the set of points lying on some propagation path to a point in the reflection cone. Then the number of distinct regions in a wavefront as the wavefront evolves is $O(m^3)$, where m is the number of walls intersected by the propagation volume (the appendix describes how this bound is obtained). The time to maintain the shadow partition is at least the number of distinct regions of the partition, i.e., $\Omega(n^3)$ in the worst case, and perhaps $\Omega(n^{3/2})$ in the typical case where m is $O(n^{1/2})$. Figure 20 gives a natural example where m is in fact close to n.

4 Beam tracing

The beam tracing algorithm is similar to the shadowing algorithm, except that it does not maintain the shadow partition. It can be given simply as

> for each reflection cone C
>   for each sample point p inside C
>   compute power of path to p.

The reflection cones can be generated by a straightforward recursive process: each wall produces a reflection cone corresponding to a single-reflection path off the wall; if C is a cone corresponding to a sequence of reflections off walls $w_1, \ldots, w_k$, then for each wall w intersecting C, there is a cone for the sequence $w_1, \ldots, w_k, w$. Section 6 describes efficient methods that enumerate both the walls and the sample points inside a reflection cone.

Given sample point p lying in a reflection cone, it is easy to find the corresponding propagation path to p. The power computation using formula (1) requires the discovery of the walls that intersect the propagation path. Section 6 describes a spatial data structure that can be used to trace a propagation path, finding the intersected walls in time that is approximately constant per wall. However, even with the spatial data structure, path tracing is the dominant cost of the algorithm. Consequently, it is useful to avoid a path power computation if it will not contribute significant power to a sample point; here significant power is defined as a fraction, say .01, of the power already accumulated at the sample point.

Let w be the last wall in the reflection sequence of a reflection cone C, and let m be an interior point of $C \cap W$, say the centroid of the vertices of $C \cap W$. Cone C is assigned an *estimated power* β_C, defined as the power from the base to m (including the final reflection coefficient off wall w).

The value β_C can be used to filter path power computations, in two ways. First, if β_C is less than an absolute threshold, say -90 dBm, then C is ignored (as are any reflection cones produced by walls intersecting C). Second, suppose β_C is above threshold and sample point p lies inside C. An upper bound estimate of the path power to p is $\beta_C(d_m/d_p)^2$, where d_m and d_p are propagation path distances to m and p, respectively. If this upper bound cannot add significant power to p, then the path is ignored. Otherwise the path is traced and path power computed; notice that β_C does not affect the actual path power computation.

It is possible that a filtered-out path in fact contributes significant power to a sample point. This might happen, for example, if there is a wide variation on the path power to points of $C \cap w$, and β_C happens to

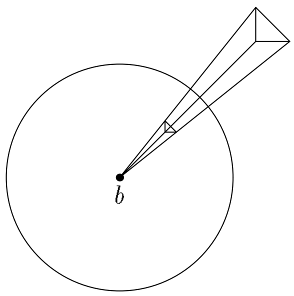


Figure 6: An initial ray is determined by a triangle on surface of sphere; b is base.

be a low estimate. The probability of such erroneously-filtered paths is minimized by restricting filtering to narrow cones, where a cone C is *narrow* if the maximum difference between any two direction vectors from the virtual transmitter of C to points in C is small, say ten degrees.

As mentioned in section 3.1, the number of reflection cones might be expected to grow as $n^{(r+1)/2}$. Similarly, the number of sample points inside a reflection cone might be expected to be about $s^{1/2}$, where s is the total number of sample points. Hence without filtering, the number of path power computations might be expected to grow as $s^{1/2}n^{(r+1)/2}$. Section 7.3 gives experimental results; filtering dramatically reduces the number of path power computations and contributes to the practical efficiency of the beam tracing algorithm.

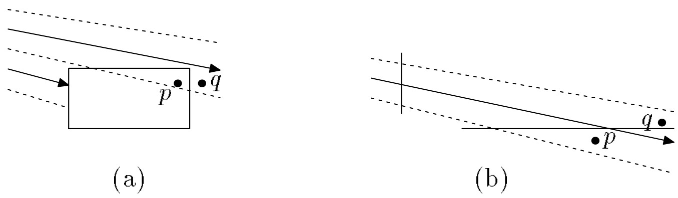
5 Ray tracing

Consider a small sphere around the base. Points on the sphere can be interpreted as direction vectors, pointing away from the base. Choose a partition of the sphere into triangles of reasonably uniform size and shape.

The ray tracing algorithm launches a sample ray for each triangle on the sphere. Each ray has a *canonical path*, which determines the path followed by the ray, and a *capture cone*, which determines the sample points captured by the ray. The canonical path follows the direction of the ray. The initial direction of the ray is the centroid of the triangle. The initial capture cone consists of all points reachable from the base in a direction that lies inside the triangle. See Figure 6.

When the canonical path intersects a wall, a reflected ray is spawned. Its direction is the reflection of the direction of the original ray in the plane of the wall. Its capture cone is the reflection of the capture cone of the original ray, truncated by the plane of the wall.

For each ray, the algorithm records the accumulated product of reflection and transmission coefficients determined by the canonical path. The algorithm can assign power to any point on the canonical path, using the accumulated product and the distance along the path to the base. If a sample point lies inside a capture cone, the algorithm assumes that there is a path to the sample point that is similar to the canonical path. The sample point is assigned the power of a nearby point on the canonical path (see below).



(a) (b)

Figure 7: Dotted lines outline capture cone. In (a), both canonical paths miss the horizontal wall. In (b), the capture cone between two wall intersections is truncated by the wall planes.

The ray tracing algorithm stops tracing a ray if it leaves the bounding box of the environment, or if the power assigned to the canonical path falls below a fixed threshold.

5.1 Sampling artifacts

The discretization of the ray tracing algorithm causes sampling artifacts. For example, Figure 7(a) depicts two adjacent rays. Neither canonical path intersects the horizontal wall, and hence no reflected ray is generated. Similarly, the shadowing of points p and q is missed.

It seems difficult to eliminate all sampling artifacts, at least while keeping the ray tracing algorithm relatively simple. Sampling artifacts can be reduced by minimizing the spacing between adjacent rays and by careful implementation. For example, the algorithm can test for sample point capture between wall intersections, with the capture cone temporarily truncated by the planes of the two walls. In Figure 7(b), q is captured before the intersection with the horizontal wall and hence its path power does not reflect the transmission coefficient of the horizontal wall; p is captured after the intersection. Of course, if the canonical path does not intersect the wall, as in Figure 7(a), the shadowing is missed.

5.2 Ray splitting

The spatial separation of adjacent rays increases linearly with the distance traveled by the rays. By choosing sufficiently many initial rays, the eventual separation could be controlled. However, it is hard to predict the maximum distance traveled by any ray, and it is undesirable to have high ray density near the base.

Ray splitting [4] allows the the separation between rays to be bounded, independently of the distance travelled by the ray. It is easy to determine spatial separation, using the angle between rays and the distance traveled by the ray. If the spatial separation exceeds a bound, the ray is split into four subrays, halving the separation of rays. See Figure 8. The original capture cone ends at a triangular face, which is partitioned into four subtriangles; each such subtriangle is the starting face of the capture cone of a subray. The original canonical path ends at the centroid of the triangular face. Though the canonical path of each subray could start at the centroid of the corresponding subtriangle, the canonical paths of ray and subray would then not be connected, and a wall intersection might be missed. Rather, the canonical path of each subray can start at the endpoint of the original path. This ensures connectivity, though the concatenated path is not straight.

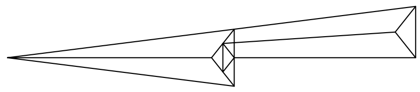


Figure 8: Ray splitting to maintain spatial separation.

The total bend due to ray splitting is at most half the initial angular separation of rays. Notice also that the canonical path of three of the four subrays lies partially outside the corresponding capture cone.

5.3 Analysis

A *ray tree* describes the branching of each ray. Each node of the tree corresponds to a segment of a canonical path, up to a wall intersection or a split. The root of the tree corresponds to the path segment leaving the base. If a path segment ends at a wall intersection, then the corresponding node has two children, one for the reflected path and one for the transmitted path; if the path segment ends because of a split, then the node has four children. A node is a leaf if the corresponding segment leaves the environment or if the power associated to the segment drops below threshold. The *ray forest* is the collection of ray trees, one for each initial ray.

Assuming that each initial ray splits at least once, the total number of leaves in a ray forest is bounded by

$$\frac{4\pi\beta_{1m}}{A\beta_t} \tag{4}$$

where β_{1m} is the free-space power at one meter, β_t is the threshold power, and A is the cross-sectional area of a ray just after a split (in meters squared).

To see this, consider a node n of the tree corresponding to a segment of a canonical path. For each point p on the segment, consider the sphere that is centered at the virtual transmitter of the segment and that passes through p. A portion of the sphere, a spherical triangle, lies inside the capture cone. The power at p decreases quadratically in the distance from the virtual transmitter, while the area of the triangle increases quadratically. Hence the product of power and area is constant over the path segment. Let $P(n)$ denote the power-area product of node n.

Now notice that the sum $\sum P(r)$ as r varies over root nodes, is just $4\pi\beta_{1m}$, since the surface area of a one-meter-radius sphere is 4π and the power at one meter is β_{1m}. Also notice that for each internal node n in the tree, the sum $\sum P(c)$, c varying over the children of n, is at most $P(n)$: at a ray-splitting node, each of the four children has one-fourth the triangular area, and at a wall-intersection node, the sum of transmission and reflection coefficients is at most 1. Hence the sum over all leaves $\sum P(l) \leq 4\pi\beta_{1m}$. Each leaf node must be assigned a value at least $\beta_t A$, since it has triangular area at least A and power at least β_t. The bound (4) follows.

The bound (4) does not hold if the power threshold is reached before a ray splits. This could happen if the initial angular separation of rays is very small, or if the power of a ray is rapidly attenuated by reflection and transmission.

For the typical parameter values $\beta_{1m} = -23$dBm, $\beta_t = -90$dBm, $A = 1$ (for a spatial separation of two meters), the number of ray forest leaves is bounded by about 63×10^6. This large bound is pessimistic for most environments, since rays leave the environment rather than die because of the power threshold.

6 Data structures

6.1 A spatial data structure

We describe a spatial data structure based on triangulations that describes the environment. It can be used to answer two geometric queries with reasonable efficiency: report the walls intersected by a query line segment, and report all walls inside a convex polyhedral query cone. The first query is used both by the beam tracing algorithm to compute path power and by the ray tracing algorithm to determine the next wall intersected by a ray. The second query is used by the beam tracing algorithm in the recursion that generates reflection cones.

To use the triangulation data structure, the allowed walls must be restricted. Each wall polygon must either be horizontal, of arbitrary shape, or a vertical rectangle, i.e., with two sides parallel to the z-axis and top and bottom horizontal. Most building walls can be modeled with polygons of this type; an exception is a cathedral ceiling, which is neither horizontal nor vertical.

Triangulations. A *triangulation* [8] is a graph embedded in the plane so that all bounded regions are triangles and so that the unbounded region has convex complement. Any set of planar line segments (that intersect only at endpoints) can be triangulated by adding enough new edges so that a triangulation results. Many algorithms are known that triangulate a set of n line segments in time $O(n \log n)$ [8].

A basic operation on triangulation is point location, which returns the triangle containing a query point. A second basic operation on a triangulation is a walk. Given segment pq and the triangle containing p, the walk reports all the edges intersecting pq in order and then reports the triangle containing q. The walk can easily be implemented in time linear in the number of reported edges.

The triangulation data structure. A *distinguished z-value* is a z-coordinate of a wall polygon vertex. The triangulation data structure consists of a set of triangulations. For each distinguished z-value, there is a triangulation of the edges of the horizontal walls at that z-value. Some triangles lie interior to a horizontal wall; each such triangle is marked with the wall that contains it. For each open interval between two adjacent distinguished z-values, there is a triangulation of the horizontal cross-section; notice that the horizontal cross-section consists of an edge for each vertical wall spanning the z-interval, and that the cross-section is constant over the interval. Each edge of the triangulation is marked with the wall that contains it, if any.

The space required to store a triangulation is linear in the number of edges that it contains. Since only linearly many edges need to be added to triangulate a set of line segments, the triangulation data structure requires space proportional to the number of horizontal walls plus $\sum n_i$, where n_i is the number of vertical walls that span the ith z-interval. In the worst case $\sum n_i$ is quadratic in the total number of vertical walls. However, most building models have only a few distinguished z-values per building floor, perhaps one for the floor, one for the false ceiling, and one for the true ceiling; and most walls span at most a single floor.

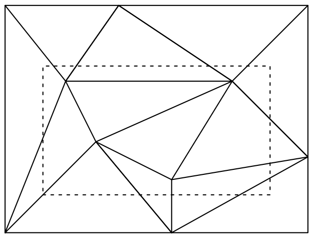


Figure 9: Query polygon (dashed) separates triangulation into inside and outside; graph search can report triangles and edges inside.

Hence in practice, the sum is linear in the number of vertical walls.

Line segment queries. The following strategy reports the walls intersected by query line segment. Split the line segment into subsegments so that each subsegment lies within a single z interval. The vertical walls intersected by a single subsegment can be determined using a walk in the corresponding triangulation, using the xy-projection of the subsegment. The horizontal wall crossed at each distinguished z-value, if any, can be discovered by a point location in the corresponding triangulation.

Polyhedral cone queries. The following strategy reports all walls intersected by a convex polyhedral query cone C; it is repeated for each triangulation that intersects the z-interval spanned by the cone.

First consider a triangulation of horizontal walls at a single distinguished z-value. The query cone intersects the horizontal plane through the z-value in a convex polygon. The algorithm walks each edge of the polygon in turn, marking the triangulation edges intersected by polygon edges. See Figure 9. The edges partially or fully inside the polygon can then be enumerated by a standard graph search, starting with the marked edges. Each triangle incident to an enumerated edge is examined to see if it is marked with a horizontal wall.

The approach is similar for a triangulation corresponding to an interval of z-values. Consider the subset of the query cone that lies within the z-interval. The orthogonal projection onto the xy-plane of this subset is a convex polygon. See Figure 10. The edges of this polygon are used for the walk, as before. In this case, the graph search examines edges of the triangulation for intersected walls, rather than triangles.

Performance. The following numbers are based on the experiments reported in section 7. For a line segment query, the number of triangulation edges traversed per reported wall varied between 2.5 and 6, depending upon the uniformity of building walls. For the polyhedral cone query, the number of triangulation edges visited per reported wall usually varied between 6 and 20, though sometimes it was more than 100.

For point location queries, a rectangular grid was superimposed on the triangulation; each grid cell stored the triangle containing the center of the cell. A query point can be located by a triangulation walk from the stored triangle of the cell containing the query point. On average, the walk traversed between .5 and 2.2 triangulation edges, indicating that this simple grid is quite adequate.

6.2 Sample point data structure

Both the beam tracing algorithm and the ray tracing algorithm require a primitive that returns the sample points inside a convex query polyhedron. Assuming that the sample points are the lattice points of a rectangular grid lying on a plane, this primitive can be implemented with polygon scan-conversion. The query polyhedron intersects the sample plane in a convex polygon. The sample points can be viewed as lying on a family of parallel lines. Each such line either misses the polygon, or intersects at two points, in which case all intermediate sample points are returned.

7 Experimental results

This section describes the results of experiments using the ray tracing and beam tracing algorithms. Both algorithms were implemented in C++ using the geometric data structures described in section 6. All times are for an SGI R4400, running at a clock-doubled rate of 100 Mhz (for an effective clock rate of 200 Mhz).

Figure 11 gives a capsule description of the buildings used in the experiments. For each building, a representative location of the base was chosen and used throughout. Unless otherwise indicated, the sample points were the lattice points of a regular grid with 2-meter spacing on a horizontal plane inside the building; the number of sample points appears in Figure 11. For all runs, the frequency was 1.92 Ghz and the transmitter power was 34 mw, giving a free-space received power of -23 dBm at one meter.

7.1 The full ray tracing algorithm.

The ray tracing algorithm was run on each building, with a power threshold of -90dBm. The spatial separation was two meters, the same as the spacing between sample points. The initial angular separation was about 4 degrees, resulting in 2048 initial rays. With these values, an initial ray splits after traveling about 30 meters; subsequent splits happen every time the distance traveled doubles.

For each building, Figure 12 reports the total number of segments in each ray forest, the total number of propagation paths to all sample points, and the average number of paths per sample point. Running times appear in the last column of Figure 13; they are roughly linear in the size of the ray forest. The most

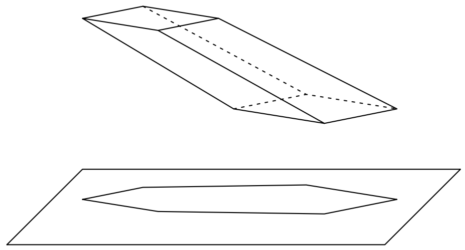


Figure 10: A query cone section projects to a convex polygon.

Building	Floors	Walls	size (m)	Sample points	Comments
Bell South	1	264	30 × 70	505	lossless walls
Crawford Hill	1	81	15 × 130	418	Figure 1.
Curved	1	277	30 × 80	659	
Hynes	4	1436	130 × 170	5240	some highly lossy walls
Indiana	2	278	120 × 250	7060	
Middletown	1	1367	200 × 200	3651	
Mt Airy	2	2316	60 × 220	2449	Figure 2.
NYSE		1596	30 × 90	523	multistory open area
Prudential	2	1175	65 × 85	1366	

Figure 11: Sample buildings.

significant run-time cost is the query that returns the sample points that lie in a section of a capture cone. This query occurs every time a ray hits the sample plane, as described in section 5.

The size of a ray forest depends upon the environment in complicated ways. For example, the NYSE is a small building with a large ray forest, because it has a multistory open area in which rays can reflect many times. Lossy walls reduce ray forest size dramatically. Bell South is the only building with strictly lossless walls; it has a much larger forest than Crawford Hill, which has similar size but lossy walls (transmission and reflection coefficients summing to at most .7). Similarly, Hynes has structural walls modeled with 26 dB transmission losses, resulting in a relatively small forest.

The number of propagation paths per sample point is large. However, for each sample point, almost all paths to the sample point contribute negligible power. For example, at a sample point in Prudential with a total received power of -44 dBm, there were two paths with power between -45 and -50 dBm, two paths between -50 and -60 dBm, 15 paths between -60 and -70 dBm, 170 paths between -70 dBm and -80 dBm, and 810 paths between -80 dBm and -90 dBm. The distribution of path powers to other sample points is probably similar.

7.2 Restricting reflections

The size of the ray forest and the execution time of the algorithm can be reduced by restricting the number of reflections allowed on any propagation path. Figure 13 gives the running times of the ray tracing algorithm as a function of the number of allowed reflections; ray forest size is correspondingly reduced.

Reducing the number of reflections always decreases the power computed at each sample point; Figure 14 gives the average difference at each sample point, taking the unbounded-reflection value as the reference value. Standard deviation values are not shown but are typically very close to the average difference value, for each number of reflections. The maximum difference is large, up to 40 dB at two reflections and 7 dB at five reflections. Clearly, for some small number of sample points, the only high-power propagation paths require several reflections.

Building	Tree size ($\times 10^6$)	Total paths ($\times 10^6$)	Ave paths ($\times 10^3$)
Bell South	13.5	9.0	17.8
Crawford Hill	.2	.1	.2
Curved	.8	.4	.5
Hynes	1.3	1.2	.2
Indiana	5.2	5.7	.8
Middletown	7.5	5.9	1.6
Mt Airy	8.7	5.7	10.9
NYSE	7.7	2.4	4.6
Prudential	4.4	2.8	2.1

Figure 12: Full ray tracing algorithm. Tree size is total over all initial rays. The last column is average number of paths per sample point.

	reflections					
Building	1	2	3	4	5	∞
Bell South	:06	:26	1:21	3:29	7:12	31:08
Crawford Hill	:04	:08	:14	:20	:25	:30
Curved	:06	:18	:35	:54	1:10	1:34
Hynes	:15	:41	1:26	2:25	3:12	4:30
Indiana	:43	2:29	5:56	10:35	15:40	25:30
Middletown	:18	1:03	2:53	6:11	11:33	29:21
Mt Airy	:19	1:06	3:13	6:55	11:40	29:01
NYSE	:14	1:02	2:41	6:23	11:06	21:16
Prudential	:14	:52	2:20	4:25	6:54	10:46

Figure 13: Running time of ray tracing algorithm, mm:ss. Initial angular separation of rays is about 4 degrees; ray separation is 2 meters.

Building	reflections 1	2	3	4	5
Bell South	7.8	5.0	3.1	1.8	1.1
Crawford Hill	2.6	1.1	.4	.2	.1
Curved	4.5	2.7	1.8	1.1	.5
Hynes	3.3	1.7	1.0	.6	.4
Indiana	4.8	3.0	1.8	1.0	.6
Middletown	9.2	5.4	2.9	1.5	.8
Mt Airy	8.3	5.3	3.4	2.3	1.4
NYSE	7.4	4.6	2.7	1.4	.7
Prudential	6.9	4.5	3.5	1.3	

Figure 14: Average difference (dB) introduced by restricting the number of reflections, with unbounded-reflection values taken as reference.

7.3 Beam tracing

The beam tracing algorithm was run on each building with the filtering described in section 4, and in some cases without the filtering. For the filtered runs, a reflection cone was ignored if its estimated power was -90 dBm or below; a propagation path to a sample point was ignored if it was estimated to contribute less than .01 times the power already accumulated at the sample point. Figure 15 gives the running times of the filtered beam tracing algorithm. For one or two reflections, more than 50% of running time is spent doing triangulation walks; for higher reflections, the manipulation of reflection cones is dominant. The unfiltered running times are not given but are much worse, up to 30 times slower in some cases. Compared with the ray tracing algorithm, the filtered beam tracing algorithm is usually better for one and two reflections, mixed for three, and usually worse for four or more reflections.

The average difference between filtered and unfiltered runs was typically 0.1 dB for a one reflection map, increasing to between 0.4 and 0.7 dB at three reflections; the standard deviation varied between 0.2 and 0.4 dB. The maximum difference was 2 to 5 dB, with larger values at higher reflections.

Figure 16 gives more details on selected runs. The total number of reflection cones increases roughly as $n^{(r+1)/2}$ as predicted in section 3.1, while the number of reflection cones with estimated power above the -90 dBm threshold is much smaller and grows more slowly. The column labeled 'paths' gives the total number of propagation paths, i.e., the number of times a sample point lies inside a reflection cone (ignoring the threshold). Particularly for the larger environments (Mt Airy and Prudential), the number of propagation paths is smaller than the estimate of $s^{1/2}n^{(r+1)/2}$ of section 4. The number of sample points per reflection cone decreases with the number of reflections, likely because the average angle subtended by a reflection cone decreases with reflections. The column labelled 'walks' gives the actual number of path power computations performed; clearly, the filtering has dramatically reduced this number from the number of propagation paths. The number of path computations per sample point gives an absolute measure of the efficacy of filtering; given that filtering does not estimate transmission loss after the last reflection, the filtering is quite effective.

	reflections				
Building	1	2	3	4	5
Bell South	:02	:08	:46	4:11	7:31
Crawford Hill	:01	:04	:10	:27	:51
Curved	:02	:09	:38	2:05	5:19
Hynes	:12	:53	2:39	6:18	14:08
Indiana	:11	:37	1:44	4:31	10:24
Middletown	:14	1:09	4:04	13:09	39:20
Mt Airy	:11	1:40	8:03	32:52	103:03
NYSE	:05	:36	6:08	11:00	
Prudential	:07	:44	4:25	19:46	66:31

Figure 15: Running time of beam tracing algorithm, mm:ss.

7.4 Sampling error

The ray tracing algorithm computes some propagation path powers incorrectly because of sampling artifacts (see section 5). The computed power can be either too large, if the shadowing of a wall is missed, or too small, if a wall is incorrectly determined to shadow a sample point.

The magnitude of the errors can be estimated using the beam tracing algorithm as a reference. For each of the runs described above with up to four reflections, the beam tracing and ray tracing coverage maps were compared. In all cases, the ray tracing maps had slightly higher average power; the average difference increased from close to 0dB at one reflection to 1 to 2dB at four reflections. (It is not clear why the average difference was not zero.) The standard deviation was between 2 and 4 dB, usually decreasing slightly as the number of reflections increased. The maximum difference was large, as much as 15 to 20 dB, though it too usually decreased with the number of reflections.

The sampling parameters used for runs of the ray tracing algorithm (two-meter ray separation, 4-degree initial angular separation) are large, and it is perhaps surprising that the ray tracing coverage maps are so close to the beam tracing maps. In fact, large systematic differences between ray tracing and beam tracing maps are possible. For example, the Middletown building is modeled with interior walls that are 2.7 meters high and end in a false ceiling; the space between false and true ceilings is an additional 1.5 meters high and is (perhaps incorrectly) modeled with no interior walls (see Figure 17). If the sample points are placed on a horizontal plane slightly below the top of the interior walls, then the ray tracing algorithm systematically predicts much higher received power for sample points far from the base. The reason appears to be that the ray tracing algorithm discovers the “propagation path” given schematically in Figure 17. This path enters the false ceiling area at a slight angle, bends slightly (see section 5.2), propagates in the false ceiling area without hitting many walls, and then captures the sample points. By contrast, any true propagation path would be attenuated by many wall intersections.

Figure 18 documents the dependence of sampling error on ray separation for a less-exceptional sample plane in Prudential (approximately a meter above the floor). A separation parameter simultaneously gives

Building	refl	cones ($\times 10^3$)	cones > -90dBm ($\times 10^3$)	paths ($\times 10^6$)	walks ($\times 10^3$)	walks/ sample	walks/ cone
Bell South	2	7	3.8	.1	35	69	9.1
Bell South	3	128	19.1	1.4	86	171	4.5
Bell South	4	1808	60.2	14.1	141	279	2.3
Crawford Hill	2	3	.6	.2	19	47	31.0
Crawford Hill	3	61	2.2	3.4	32	78	14.2
Crawford Hill	4	1269	4.5	57.2	41	98	9.1
Mt Airy	2	131	19.1	1.2	230	102	12.0
Mt Airy	3	4873	110.6	23.1	614	273	5.5
Mt Airy	4	155212	407.8	436.5	1213	539	3.0
Prudential	2	54	18.3	.7	114	84	6.2
Prudential	3	1434	105.9	11.9	276	202	2.6
Prudential	4	30886	363.8	189.9	537	393	1.5

Figure 16: Beam tracing: reflection cones, total and with estimated power above threshold; number of propagation paths; number of path power computations ('walks') performed; and path power computations per sample point and per cone.

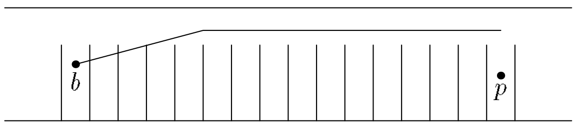


Figure 17: Elevation view of Middletown. Ray path from b bends slightly after entering false ceiling area and eventually captures sample point p.

the grid spacing and ray separation in meters; the initial angular separation in degrees was twice as large. As the separation is reduced, the coverage maps produced by the two algorithms appear to converge. However, even at small separation, there are sample points which are assigned quite different power values. The runtime of the both algorithms grows roughly as the inverse square of the separation: in the case of the beam tracing algorithm, because the number of sample points grows at that rate, and in the case of the ray tracing algorithm, because the number of rays grows at that rate.

7.5 Threshold

The power threshold of either algorithm in effect determines an allowed propagation path loss, below which a propagation path is deemed insignificant. This allowed path loss is dependent both upon receiver sensitivity and transmitted power, and hence is system-specific. The running time of both algorithms depends upon

	Grid spacing and ray separation (m)				
	4	2	1	.5	.25
Average difference (dB)	3.9	.9	1.0	.7	.6
Standard deviation (dB)	5.5	2.6	2.0	1.4	1.0
Maximum difference	25.1	14.2	12.5	14.1	12.6
Time (beam tracing)	:15	:44	1:46	6:02	22:44
Time (ray tracing)	:27	:52	3:53	18:01	106:27

Figure 18: Difference between ray tracing and beam tracing coverage maps, as a function of separation (two reflections at Prudential). Initial angular ray separation in degrees is twice ray separation in meters.

		Threshold (dBm)				
		−70	−80	−90	−100	−110
Hynes	BT	1:24	1:56	2:39	3:07	3:47
Hynes	RT	:14	:37	1:27	2:43	4:26
Crawford Hill	BT	:07	:08	:10	:12	:22
Crawford Hill	RT	:06	:08	:14	:26	:43
Middletown	BT	2:28	3:38	4:04	4:58	6:11
Middletown	RT	:23	1:02	2:02	3:36	5:53
Mt Airy	BT	3:38	5:30	7:54	10:46	13:42
Mt Airy	RT	:24	1:02	2:12	4:37	8:18

Figure 19: Effect of beam threshold on the beam tracing (BT) algorithm and ray threshold on the ray tracing (RT) tracing; all runs include three reflections.

allowed path loss, which determines the size of the ray tree in the ray tracing algorithm and the number of significant reflection cones in the beam tracing algorithm. Figure 19 documents the sensitivity of the running time of the two algorithms to threshold, for three-reflection coverage maps at selected buildings. The ray tracing algorithm appears to be more sensitive to threshold than the beam tracing algorithm.

7.6 Summary

The experiments reported above do not indicate that either algorithm is definitively superior. For a small number of reflections, beam tracing is competitive with ray tracing in running time (and sometimes better), largely as a consequence of the extensive filtering of propagation paths. As the number of reflections increases, the manipulation of reflection cones becomes the dominant cost, and the simpler manipulation of rays gives the advantage to the ray tracing algorithm. The beam tracing algorithm does have the advantage of having fewer sampling artifacts.

The experiments reported in section 7.2 suggest that limiting predictions to two or three reflections introduces only a few dB of error. This error bound estimate is statistical, in the precise sense that it is averaged over many sample points and in the vague sense that most buildings are similar to ones considered here. Indeed, neither algorithm guarantees that the power computed at a particular sample point is anywhere close to the power defined by equations (2) and (3) of section 3. However, the model in section 3 is itself a considerable simplification. The error between model and the true, rather complex propagation of radio is likely much larger than the error resulting from incomplete simulation of the model.

8 Acknowledgements

Thanks to Brian Kernighan, Margaret Wright, and Reinaldo Valenzuela for helpful comments.

References

[1] J. B. Allen, D. A. Berkley, Image method for efficiently simulating small-room acoustics, *J. Acous. Soc. Amer*, Vol 65, 943–950, 1979.

[2] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, 1987.

[3] S. J. Fortune, D. M. Gay, B. W. Kernighan, O. Landron, R. A. Valenzuela, M. H. Wright, WISE design of indoor wireless systems: practical computation and optimization. *IEEE Computational Science and Engineering*, Spring 1995, pp. 58–68.

[4] P. Kreuzgruber, P. Unterberger, R. Gahleitner, A ray splitting model for indoor radio propagation associated with complex geometries, *Proceedings of the 1993 43rd IEEE Vehicular Technology Conference*, IEEE, 1993, pp. 227–230.

[5] P. Kreuzgruber, T. Bründl, W. Kuran, R. Gahleitner, Prediction of indoor radio propagation with the ray splitting model including edge diffraction and rough surfaces, *Proceedings of the 1994 IEEE 44th Vehicular Technology Conference*, IEEE, 1994, 878–882.

[6] J.W. McKown, R.L. Hamilton, Jr. Ray tracing as design tool for radio networks, *IEEE Network Magazine*, Vol 5, pp. 27–30, 1991.

[7] J.D. Parsons, *The Mobile Radio Propagation Channel*, John Wiley and sons, 1992.

[8] F. P. Preparata, M.I. Shamos, *Computational Geometry*, Springer Verlag, 1985.

[9] A. Rajkumar, B. Naylor, F. Fiesullin, L. Rogers, Predicting RF coverage in large environments using ray-beam tracing and partitioning tree represented geometry, Technical Memorandum, AT&T Bell Laboratories, 1995.

[10] S. Ramo, J.R. Whinnery, T. Van Duzer, *Fields and Waves in Communication Electronics*, 3rd edition, John Wiley and Sons, 1993.

[11] K.R. Schaubach, N.J. Davis, T.S. Rappaport, A ray tracing method for predicting path loss and delay spread in microcellular environments, *Proceedings of the 42nd IEEE Vehicular Technology Conference*, May 1993, pp. 261–264.

[12] S. Seidel, T. Rappaport, A ray tracing technique to predict path loss and delay spread inside buildings, *IEEE Globecom 92*, pp. 649–653, 1992.

[13] S. Takahashi, K. Ishida, H. Yoshiura, A. Nakagoshi, An evaluation point culling algorithm for radio propagation simulation based on the imaging method, *Virginia Tech Fifth Symposium on Wireless Personal Communications*, pp. 13.1 – 13.11, 1995.

[14] R. A. Valenzuela, A ray-tracing approach to predicting indoor wireless transmission, *Proc. 1993 IEEE 43rd Vehicular Technology Conference*, IEEE Press, Piscataway, NJ, 1993, pp. 214–218.

[15] A. Watt, M. Watt, *Advanced Animation and Rendering Techniques: Theory and Practice*, Addison-Wesley, 1992.

A Appendix: shadowing

We analyze the combinatorial complexity of a wavefront in the shadowing algorithm, justifying the bounds in section 3.1. Fix a reflection cone C, defined by reflection off a sequence of walls $w_1, \ldots, w_r$. We assume that both the number of reflections r is constant and that there is a constant bound on the number of sides of any wall; it follows that the number of faces of C is constant. Let W_C be the walls that intersect the propagation volume of C, and let m be the size of W_C. We show that at any time the shadow partition can be represented as a planar graph with $O(m^2)$ edges and regions; furthermore, as the wavefront evolves, the total number of distinct regions in the graph is $O(m^3)$.

The *shadow* of a wall w at position i is the set of points on the wavefront whose corresponding propagation paths pass through w between reflections i and $i+1$. A wall shadow is bounded by a spherical polygon, i.e., a polygon on the wavefront whose edges are segments of a diameter. Each edge of the wall shadow is either the shadow of an edge of w or lies on one of the planes bounding the reflection cone. Hence the wall shadow has a constant number of edges. The regions of the shadow partition can be obtained by overlaying all wall shadows. Since there are a constant number of wall shadows per wall, there are $O(m)$ wall shadows, and the shadow partition can be represented as a planar graph with $O(m^2)$ edges.

Consider a particular edge of a wall shadow on the wavefront. As the wavefront evolves, all points of the edge lie on a common plane. Consider all planes obtained in this way, together with each plane through a wall in W_C. Since there are $O(m)$ such planes, the arrangement [2] of the planes has at most $O(m^3)$ cells. For any cell of the arrangement lying in the reflection cone, the shadow function is invariant over the cell. Hence the number of distinct regions of the shadow partition is at most $O(m^3)$.

Figure 20 depicts Crawford Hill from a viewpoint slightly outside the left end of the building (Figure 1). This gives a natural example where the number of shadow regions is cubic in the number of building walls.

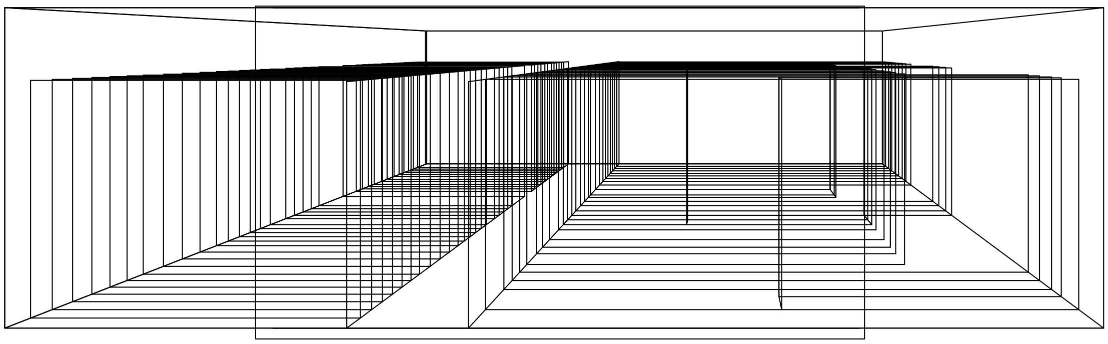


Figure 20: View of Crawford Hill from a point slightly outside the left end of figure 1.