Ray Tracing with Meta-Hierarchies

James Arvo

Apollo Systems Division of Hewlett-Packard
300 Apollo Drive
Chelmsford, MA 01824

arvo@apollo.hp.com, arvo@yale.edu

Abstract

Because ray tracing acceleration techniques differ in their relative efficiency from scene to scene, no single technique can be optimal in all circumstances. This has lead researchers to develop hybrid algorithms that combine two or more distinct techniques into one. In this paper we describe a generalization of this idea, the meta-hierarchy, which arises from a procedural encapsulation of acceleration techniques described in [Kirk 88]. We discuss the pros and cons of this mechanism and suggest future research directions.

1 Introduction

Ray tracing acceleration has received much attention in the decade since Whitted introduced his illumination model and established ray tracing as one of the preeminent algorithms for high fidelity image synthesis [Whitted 80]. Though virtually every aspect of the algorithm has been studied for potential optimization, perhaps the greatest emphasis has been on the problem of efficiently finding the first point of intersection between a ray and a complex environment. Following the taxonomy outlined in [Arvo 89] we can classify the many algorithms devised for this type of optimization into three major categories: bounding volume hierarchies, spatial subdivision, and directional techniques.

Representative algorithms in the bounding volume category are bounding box hierarchies [Rubin80] and bounding slabs [Kay 86]. Spatial subdivision algorithms include octrees [Glassner 84], BSP trees [Kaplan 87], uniform grids [Fujimoto 86], and macro regions [Devillers 89]. Directional techniques include the light buffer [Haines 86], the ray coherence algorithm [Ohta 87], and ray classification [Arvo 87]. Each dictates a particular organization of the geometric entities comprising the environment and each prescribes a specific means of processing the data to locate the closest point of ray intersection. Though there are many similarities among the algorithms of any category, the assortment of data structures that have been studied is wide, and the methods of processing these data structures are more numerous still.

Why are there so many approaches? One reason is the drive toward greater and greater efficiency, and another is the need to handle increasing scene complexity. Unfortunately there does not exist a clearly established precedence among the methods of acceleration in terms of their effectiveness. This is not merely an artifact of the difficulty of quantitative comparison. At a deeper level it can be attributed to the fact that each method presents a different set of strengths and weaknesses and can potentially out pace all the others on selected scenes. For example, uniform spatial subdivision can be expected to do exceptionally well with geometric primitives dispersed somewhat uniformly in space while the simplicity of bounding volume hierarchies make them very effective for environments consisting of a small number of primitives. Moreover, ray classification may gain the edge in scenes containing tessellated solids due to special optimizations such as back-face culling.

Factors such as these have lead many researchers to develop hybrid algorithms that capitalize on the strengths of several different basic techniques simultaneously, recognizing that no single approach is optimal for all scenes. We will briefly discuss the notion of a hybrid then introduce the *meta-hierarchy*, a general mechanism for combining acceleration techniques.

2 Hybrids

For this discussion a *hybrid acceleration technique* will mean an algorithm that employs several distinct ray intersection techniques in the rendering of a single scene. The particular technique invoked may vary during the processing of each ray according to the location of the objects tested or, as another possibility, the technique may be selected based on the type of ray: originating at the eye, a light source, or from a surface.

Two popular approaches that have been combined into hybrids are spatial subdivision and bounding volume hierarchies. Scherson and Caspary performed empirical studies in order to determine conditions in which one or the other proved more efficient [Scherson 87]. Their results led them to a hybrid that employed coarse octree subdivision at the top level and bounding volume hierarchies in the leaf voxels.

Synder and Barr took a similar approach but performed the initial spatial subdivision using a uniform grid structure [Snyder 87]. The additional advantage of instancing the same leaf node structure from many voxels allowed them to create scenes of enormous complexity.

Glassner combined spatial subdivision with bounding volume hierarchies in a different way, creating a pure bounding volume hierarchy that possesses some of the characteristics of spatial subdivision [Glassner 88]. In particular, the bounding volumes are formed in such a way that siblings are guaranteed to be disjoint, in direct analogy with the voxels of spatial subdivision. This is done by using octree subdivision to guide the creation of the volumes.

Finally, Devillers [Devillers 89] introduced the notion of nesting one uniform grid structure within the voxels of another. Though this is not a heterogeneous mix of techniques as those discussed above, it nevertheless captures the essence of a hybrid.

3 Aggregate Objects

It is a common observation among ray tracing practitioners that the different primitive objects comprising an environment – spheres, polygons, quadrics, parametric surfaces, etc. – can be encapsulated with a uniform procedural interface, or data abstraction, providing a clean and extensible implementation. Such an abstraction defines an *object* that is capable of performing the fundamental operations required in ray tracing: ray intersection, computation of surface normals, computation of bounding volumes, etc. Even without the aid of an object-oriented programming language this is easily accomplished with little or no degradation in performance. A common trick is to represent the object as a collection of function pointers, each responsible for performing one of the fundamental operations for the particular type of geometric primitive.

This approach can be extended slightly to yield a convenient means of encapsulating acceleration techniques as well. This entails representing acceleration algorithms as *aggregate objects*, each with a procedural interface identical to that of primitive objects with the exception of one additional operation, *add_child* (See [Kirk 88], reprinted here). This operation makes an instance of a primitive or aggregate object subordinate to another aggregate object. The *parent* object is then responsible for performing the ray intersection operation on its collection of subordinate objects, or *children*, using their associated ray intersection methods. This is the very operation that most acceleration techniques are designed to perform efficiently.

Encapsulating an acceleration technique as another form of object is quite useful. It makes a ray tracer easy to extend with new primitive objects or acceleration techniques, and by virtue of the interchangeability of aggregate objects, it allows one to easily alternate between several acceleration techniques within the same ray tracer for testing, benchmarking, and development purposes. Different acceleration techniques may be used as interchangeably as primitive objects typically are in the context of ray tracing. More importantly, however, it provides a new way to construct and efficiently render complex environments.

4 Meta-Hierarchies

Rather than creating a hybrid by combining two or more specific acceleration techniques in a fixed way, a more general approach is to construct a *meta-hierarchy* of acceleration techniques. One of the important advantages of the object data abstraction described in [Kirk 88] is that it lends itself to this type of construction. The uniformity of the interface across primitive and aggregate objects implies that any object can be the child of any aggregate object. This allows us to nest acceleration techniques within one another to any depth, resulting in a higher-level hierarchy. An aggregate object nested within another appears to its parent as a bounding volume and an associated intersection method for its contents. The parent object therefore deals with it in exactly the same way it would a primitive object. Because the aggregate objects forming the nodes of this hierarchy may themselves employ hierarchical data structures we refer to the top-level hierarchy as a meta-hierarchy.

Through meta-hierarchies we can easily create a large number of hybrid acceleration techniques. Assuming we have encapsulated implementations of all the relevant algorithms, using this technique we can nest bounding volume hierarchies or uniform grids within a spatial subdivision technique, as in the hybrids of section 2. We can also choose to nest several different techniques simultaneously within a single structure such as an octree, employing uniform grids for collections of primitives that are fairly evenly dispersed and bounding volume hierarchies for small but closely packed collections of primitives. See [Kirk 88] for a specific example of how this type of nesting was used in the production of a large-scale ray traced animation.

A single aggregate object may be instanced multiple times within the same environment in exact analogy with canonical primitive objects. As observed in [Snyder 87], this allows us to save space by reusing not only the objects managed by the aggregate but also the internal data structures (e.g. octrees or grids). We also save on the time required to build the aggregate initially since it is only built once. In the case of acceleration techniques that are constructed by lazy evaluation [Arvo 87], their data structures may be partially built on behalf of each instance as rays encounter them. This automatically benefits all other instances.

5 Drawbacks

Because a meta-hierarchy does not constrain the nesting of acceleration techniques, it is quite amenable to experimentation with new hybrids, allowing the creation of novel combinations on a scene by scene basis. This flexibility has its price, however, so a hybrid constructed by means of a meta-hierarchy can potentially be less efficient than a specially designed hybrid of the same overall structure. For example, the data abstraction at the heart of the meta-hierarchy imposes a mandatory layer of indirect function invocations between the levels of the hierarchy. This can sometimes be streamlined if we know *a priori* that method X will be nested within method Y and we can dispense with the generality of function pointers, factor out common expressions, etc.

Another potential drawback is that the fixed procedural interface may preclude the communication of useful information between the nodes of the hierarchy. In particular, because an aggregate object treats each of its children as though it were a primitive object, it does not differentiate between its children's children. If an aggregate is nested within an octree, for example, the entire collection of objects will be treated as a candidate for intersection in each voxel it penetrates. It is therefore crucial that spatial subdivision techniques guard against multiple tests of the same object per ray. Failing to do so can be extremely costly if the object in question happens to be a complex aggregate.

6 Future Directions

Though the flexibility of the meta-hierarchy approach can be a great asset, it also presents challenges in terms of selecting the appropriate collections of objects to bundle as aggregate objects, choosing the acceleration techniques to apply to these groups, and deciding the level of nesting. Currently these decisions must be made by the user. This can be a fairly complicated process because the tradeoffs are not always obvious and can even be counter-intuitive. A costly acceleration technique may be more effective than an inexpensive one if, on average, it eliminates a few more unnecessary intersection tests with complex primitives. Furthermore, tradeoffs between space and time can have unexpected results when paging costs are incurred. Considerations such as these are important because poor choices can easily increase the running time instead of decrease it. It is natural to ask whether this process can be performed automatically.

Goldsmith and Salmon addressed the closely related problem of automatically creating efficient bounding volume hierarchies [Goldsmith 87]. This can be viewed as a special case of a meta-hierarchy: one in which the nodes encapsulate exhaustive search coupled with bounding volume tests.[1] Their approach was based on a metric for predicting the expected cost of intersecting a random ray with a given hierarchy. The key was to examine the conditional probability that a ray will intersect a volume given that it intersects a surrounding volume. Because this concept is quite powerful and more generally applicable, we discuss it in detail in appendix A. Goldsmith et al. also introduced a hierarchy creation scheme designed to produce hierarchies that nearly minimize this metric. It may be possible to extend this approach to meta-hierarchies.

Naturally, in the context of meta-hierarchies there are additional factors to take into consideration. Each internal node is an aggregate object, a black box that can intersect any given ray with its set of child objects with a certain average cost. As such, the nodes of the hierarchy can be very heterogeneous, dif-

[1] However, it is not suggested that a meta-hierarchy be used to implement a pure bounding volume hierarchy in this way.

fering not only in size and shape, but also average cost in terms of space and time for both initial creation and subsequent use. These costs can depend quite strongly on the spatial arrangement and characteristics of the child objects, and each type of acceleration technique (encapsulated as an aggregate object) presents different tradeoffs. Estimating the expected costs for each acceleration technique is quite difficult and remains an open problem. It is quite necessary, however, if we are to formulate a useful *a priori* metric for the effectiveness of the entire meta-hierarchy. But even with such a metric, finding a near-optimal meta-hierarchy is a very difficult combinatorial optimization problem.

A technique which may be applicable in this context is *simulated annealing* [Kirkpatrick 83]. This is a strategy for probabilistically exploring the solution space of large combinatorial optimization problems and finding incremental improvements in the objective function without getting stuck in a poor local minimum. It is closely related to principles in thermodynamics, and was originally motivated by nature's ability to find near-optimal solutions to problems of vast complexity, such as aligning the molecules of a liquid into a near-minimum-energy configuration as they freeze into a crystal. It has been fairly successful at "solving" NP-hard problems such as the traveling salesman and chip placement.

To apply this technique we think of the objective function that we want to minimize as the "energy" of the system, and we introduce a notion of "temperature" that is gradually dropped during the optimization process. We produce random perturbations in the configuration and then decide whether or not to keep the result based on the change in energy and the current temperature. A perturbation of a meta-hierarchy may entail moving an object from one aggregate to another, substituting one acceleration technique for another, or splitting one aggregate into two. If the algorithm were "greedy" and kept only the new configurations that lowered the energy (i.e. expected cost) at each step, it would descend immediately into a local minimum. This may not be of much value if the starting configuration is far from optimal. On the other hand, the simulated annealing approach accepts a new configuration with probability $\exp\left(-\frac{\Delta E}{kT}\right)$, where ΔE is the change in energy, and T is the current temperature. This allows the search process to encompass configurations that differ so greatly from one another that many apparently counter-productive changes are required to move between them. As the temperature is lowered, the algorithm becomes more greedy. The all-important strategy for randomization and lowering of temperature is called the *annealing schedule* and is extremely problem dependent. With the appropriate annealing schedule this technique may provide a simple means of coping with the combinatorial explosion of possibilities presented by meta-hierarchies.

7 Conclusion

By encapsulating acceleration techniques as aggregate objects and combining them into meta-hierarchies, we can easily explore a wide range of new hybrid techniques. The automatic construction of near-optimal meta-hierarchies is an open problem, though existing techniques for optimizing bounding volume hierarchies provide a reasonable starting point. The method of simulated annealing may prove to be useful in coping with the resulting combinatorial optimization problem.

Appendix A

There are several references which contain probabilistic cost analysis of acceleration techniques. See [Goldsmith 87] or [Arvo 89] for a method of estimating the expected cost of intersecting a random ray with a bounding volume hierarchy. For expected costs associated with spatial subdivision see [Devillers 89] and the references therein. Given the potential importance of this type of estimation to the problem of automatic meta-hierarchy creation, we now turn to a conditional probability theorem which forms the basis of the analysis.

Theorem 1 *Assuming a uniform distribution of rays* [2]*, the conditional probability that a ray will intersect a convex bounding volume $\mathcal{B}$ given that it intersects a surrounding convex bounding volume $\mathcal{A}$ is $S(\mathcal{B})/S(\mathcal{A})$.*

Note that this theorem holds for all shapes and relative orientations of bounding volumes provided that both $\mathcal{A}$ and $\mathcal{B}$ are convex, $\mathcal{B} \subseteq \mathcal{A}$, and the rays are uniformly distributed. Because it may be counter-intuitive that this relationship should depend only on surface areas, we outline a proof.

Let $\mathcal{A}$ and $\mathcal{B}$ be nested convex volumes and let r be a random ray hitting volume $\mathcal{A}$, that is, $r \cap \mathcal{A}$. In order to find $\Pr(\, r \cap \mathcal{B}\,)$ let's begin by considering only rays that are parallel to a given vector, ν. Thus

[2]As a practical definition of this, consider ray origins that are uniformly distributed within an large enclosing sphere and ray direction vectors that are uniformly and independently distributed over the unit sphere.

constrained, the probability that r hits $\mathcal{B}$ is the ratio of the projected area of $\mathcal{B}$ to the projected area of $\mathcal{A}$ in the direction ν. That is

$$\Pr(\ r \cap \mathcal{B} \,|\, r \cap \mathcal{A} \text{ and } r \parallel \nu\ ) \ = \frac{P_\nu(\mathcal{B})}{P_\nu(\mathcal{A})}. \tag{1}$$

Here $r \parallel \nu$ means that the ray r is parallel to the vector ν, and $P_\nu(\mathcal{X})$ denotes the projected area of $\mathcal{X} \subset R^3$ in the direction ν (i.e. the area of the projection of $\mathcal{X}$ onto a plane orthogonal to ν). Henceforth we shall omit the condition $r \cap \mathcal{A}$ for brevity, but it is always to be assumed that r hits $\mathcal{A}$.

If A and B are events such that $B \subseteq A$ and $A_1 \ldots A_n$ is a partitioning of A, then the rule of elimination states that

$$\Pr(\ B\ ) \ = \sum_{i=1}^{n} \Pr(\ B \,|\, A_i\ )\ \Pr(\ A_i\ )\ . \tag{2}$$

We will apply a similar principle to equation 1 in order to integrate over the condition $r \parallel \nu$, thus events $A_1 \ldots A_n$ will be replaced by the set of all directions ν in Ω, the unit sphere. Because Ω is a continuum, we must replace the probability $\Pr(\ A_i\ )$ in equation 2 with a probability density function giving the density of rays parallel to ν, and the summation must become integration over Ω. Denoting the probability density function as $f(\nu)$ we have

$$\Pr(\ r \cap \mathcal{B}\ ) \ = \int_\Omega \Pr(\ r \cap \mathcal{B} \,|\, r \parallel \nu\ )\ f(\nu)\, d\nu. \tag{3}$$

But the density of rays parallel to ν is a function of the projected area of $\mathcal{A}$ in that direction. Roughly speaking, the larger the projected area of $\mathcal{A}$ for a given direction, the more likely r is to be parallel to that direction. We therefore arrive at the following.

$$f(\nu) = \frac{P_\nu(\mathcal{A})}{\int_\Omega P_\nu(\mathcal{A})\, d\nu}. \tag{4}$$

Substituting equations 1 and 4 into equation 3 we have

$$\Pr(\ r \cap \mathcal{B}\ ) \ = \frac{\int_\Omega P_\nu(\mathcal{B})\, d\nu}{\int_\Omega P_\nu(\mathcal{A})\, d\nu} \tag{5}$$

which we can also expressed as

$$\frac{< P_\nu(\mathcal{B}) >}{< P_\nu(\mathcal{A}) >}. \tag{6}$$

Here we have used the $< \cdot >$ notation to indicate averaging over all directions ν. That is

$$< P_\nu(\mathcal{X}) > = \frac{\int_\Omega P_\nu(\mathcal{X})\, d\nu}{\int_\Omega d\nu}. \tag{7}$$

Equation 6 may not appear to be of any practical value at first, but there is a simple relationship which converts this expression to the desired one. It expresses an apparently cumbersome quantity, average projected area, in terms of something which is typically very easy to compute, namely surface area.

Theorem 2 *The average projected area of a convex set is one quarter its surface area.*

Using this fact, the expression in 6 becomes the ratio of surface areas. Therefore, if r is a random ray which is known to hit a convex bounding volume $\mathcal{A}$, then

$$\Pr(\ r \cap \mathcal{B}\ ) \ = \frac{S(\mathcal{B})}{S(\mathcal{A})} \tag{8}$$

where $\mathcal{B}$ is any convex object contained within $\mathcal{A}$. This proves theorem 1.

Note that theorem 2 is true for all convex sets independent of shape; a surprising fact which does not appear to be widely known. As van de Hulst observes, “This, apparently, is one of those theorems which gets discovered and rediscovered many times” [van de Hulst 81]. It may seem implausible that a sphere of radius one, or surface area 4π, has the same average projected area as a square with edge $\sqrt{2\pi}$, or about 2.507 (two sides, each of area 2π). An easy way to prove the theorem is to begin with a simpler case; that of a planar figure.

Theorem 3 *The average projected area of an arbitrary planar figure is one half its area.*

Proof: let $\mathcal{F}$ be a planar figure of area α and let η be a unit vector normal to $\mathcal{F}$. Then for any direction ν, the projected area of $\mathcal{F}$ in the direction of ν is given by

$$P_\nu(\mathcal{F}) = \alpha |\eta \cdot \nu|. \tag{9}$$

In order to compute $< P_\nu(\mathcal{F}) >$ we will make use of two observations. First, we can assume without loss of generality that figure $\mathcal{F}$ lies in the XY plane. Secondly, to avoid the absolute value in equation 9 we can assume that η is the positive Z-axis and integrate over the positive hemisphere. By symmetry the average value of P_ν over the hemisphere is the same as over the entire sphere.

Let ϕ be the angle between the direction vector ν and the positive Z-axis, and let θ be its rotation angle measured in the XY plane. Then a differential patch on the surface of the unit sphere is given by $\sin\phi \, d\theta \, d\phi$. Observe that when this is integrated as in equation 10, we obtain the surface area of the hemisphere.

$$\int_0^{\pi/2} \int_0^{2\pi} \sin\phi \, d\theta \, d\phi = 2\pi \tag{10}$$

Using spherical coordinates we can write the projected area function P as

$$P_\phi(\mathcal{F}) = A(\mathcal{F}) \cos\phi. \tag{11}$$

Note that the projected area is independent of θ. Averaging this function over the hemisphere we have

$$\frac{1}{2\pi} \int_0^{\pi/2} \int_0^{2\pi} A(\mathcal{F}) \cos\phi \sin\phi \, d\theta \, d\phi. \tag{12}$$

The value of this definite integral is $A(\mathcal{F})/2$, which proves theorem 3.

Now, any continuous surface can be approximated arbitrarily closely by a polygonal mesh. Let $\mathcal{X}$ be such a surface and let $x_1 \ldots x_n$ be a polygonal mesh approximating it. If $\mathcal{X}$ is convex then every point of every projection is accounted for twice – once by a point on the "front-facing" surface and once by a point on the "back-facing" surface. Using this fact we have

$$< P_\nu(\mathcal{X}) > \approx \frac{1}{2} \sum_{i=1}^{n} < P_\nu(x_i) > . \tag{13}$$

But $< P_\nu(x_i) >$ is simply one half the area of x_i, by theorem 3, so the right hand side of equation 13 is one quarter the sum of the areas of $x_1 \ldots x_n$. In the limit of the polygonal mesh approximation, this becomes one quarter the surface area of $\mathcal{X}$, and the approximation becomes equality. This proves theorem 2.

References

[Arvo 87] Arvo, James, and David Kirk, "Fast Ray Tracing by Ray Classification," Computer Graphics, Vol. 21, No. 4, July 1987, pp. 55-64.

[Arvo 89] Arvo, James, and David Kirk, "A Survey of Ray Tracing Acceleration Techniques," in "An Introduction to Ray Tracing," edited by Andrew Glassner, Academic Press, New York, 1989.

[Devillers 89] Devillers, Oliver, "The Macro-regions: an Efficient Space Subdivision Structure for Ray Tracing," Eurographics '89, W. Hansmann, F.R.A. Hopgood, W. Strasser (Editors), Elsevier Science Publishers B. V. (North-Holland), 1989, pp. 27-38.

[Fujimoto 86] Fujimoto, Akira, Takayuki Tanaka, and Kansei Iwata, "ARTS: Accelerated Ray-Tracing System," IEEE Computer Graphics and Applications, Vol. 6, No. 4, April 1986, pp. 16-26.

[Glassner 84] Glassner, Andrew S., "Space Subdivision for Fast Ray Tracing," IEEE Computer Graphics and Applications, Vol. 4, No. 10, October 1984, pp. 15-22.

[Glassner 88] Glassner, Andrew S., "Spacetime Ray Tracing for Animation," IEEE Computer Graphics and Applications, Vol. 8, No. 3, March 1988, pp. 60-70.

[Goldsmith 87] Goldsmith, Jeffrey, and J. Salmon, "Automatic Creation of Object Hierarchies for Ray Tracing," IEEE Computer Graphics and Applications, Vol. 7, No. 5, May 1987, pp. 14-20.

[Haines 86] Haines, Eric A., and Donald P. Greenberg, "The Light Buffer: A Shadow-Testing Accelerator," IEEE Computer Graphics and Applications, Vol. 6, No. 9, September 1986, pp. 6-16.

[Kaplan 87] Kaplan, Michael R., "The Use of Spatial Coherence in Ray Tracing," in Techniques for Computer Graphics, ed. David F. Rogers and Rae A. Earnshaw, Springer-Verlag, New York, 1987.

[Kay 86] Kay, Timothy L., and James Kajiya., "Ray Tracing Complex Scenes," Computer Graphics, Vol. 20, No. 4, August 1986, pp. 269-278.

[Kirk 88] Kirk, David, and James Arvo, "The Ray Tracing Kernel," Proceedings of Ausgraph '88, Melbourne, Australia, July 1988, pp. 75-82.

[Kirkpatrick 83] Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, May 13, 1983, pp. 671-680.

[Ohta 87] Ohta, Masataka, and Mamouru Maekawa, "Ray Coherence Theorem and Constant Time Ray Tracing Algorithm," Computer Graphics 1987 (Proceedings of CG International '87), ed. T. L. Kunii, pp. 303-314.

[Rubin80] Rubin, Steve, and Turner Whitted, "A Three-Dimensional Representation for Fast Rendering of Complex Scenes," Computer Graphics, Vol. 14, No. 3, July 1980, pp. 110-116.

[Scherson 87] Scherson, Isaac D., and Elisha Caspary, "Data structures and the time complexity of ray tracing," The Visual Computer, Springer-Verlag, Vol. 3, 1987, pp. 201-213.

[Snyder 87] Snyder, John M. and Alan H. Barr, "Ray Tracing Complex Models Containing Surface Tessellations," Computer Graphics, Vol. 21, No. 4, July 1987, pp. 119-126.

[van de Hulst 81] van de Hulst, H. C., "Light Scattering by Small Particles," Dover Publications, Inc., New York, 1981.

[Whitted 80] Whitted, Turner, "An Improved Illumination Model for Shaded Display," *Communications of the ACM,* 32(6), June 1980, pp. 343-349.