Abstract

We propose a method to generate penumbrae (half shadows) in scenes that are composed of simple 3-D geometric primitives. Penumbrae occur when objects are illuminated by a light source with finite extent. Traditional methods to generate penumbrae are based either on extensive geometric reasoning or on signal processing (sampling and smoothing). In either way they have a high computational complexity. We propose a third alternative that is based on function evaluation, not unlike the way in which implicit surfaces are defined by means of evaluating suitable field functions. Our method starts from a physically correct model for the shape of penumbrae, and proposes some ways to compose the penumbrae for individual geometric primitives. We discuss the efficiency issues related to our method, and we indicate some restrictions for its application.

Soft Shadows with Soft Objects: a proof of concept

Kees van Overveld
Philips Research and Department of Mathematics and Computing Science
Eindhoven University of Technology*

Mark Tigges and BrianWyvill
Department of Computer Science
University of Calgary†

September 5, 1999

Keywords shadows, penumbrae, implicit surfaces

1 Introduction

In order to improve the quality of rendered images of 3-D scenes, shadows should be taken into account. Shadows that are cast onto a surface S can take the form of *umbrae*, i.e. regions in S where no part of the light source can be seen, and *penumbrae*, i.e. regions in S where part of the light source can be seen. Regions in S where all of the light source can be seen are not in the shadow. If the extent of the light source is negligable, there is no penumbra. This is a common assumption in computer graphics for efficiency reasons.

In order to model penumbrae, two basically different approaches have been followed:

- A geometric approach: based on the notion of shadow volumes, used to model umbrae (as first proposed by Crow in 1977, [**?**]), one can

*Address: P.O.Box 513, 5600 MB, Eindhoven, The Netherlands. Email: wsinkvo@info.win.tue.nl

†Address: 2500 University Drive N.W., Calgary, Alberta,Canada, T2N 1N4 Email: [mtigges|blob]@cpsc.ucalgary.ca

apply geometric operations like Minkowsky sums or Boolean operations in order to construct polyhedral shadow volumes that also have penumbrae components ([?],[?],[?]).

- A signal processing approach: the amount of light on any given point of S is really a surface integral over a continuous light distribution function. This continuous function can be sampled in the context of ray tracing (so called distributed ray tracing, [?]) or in the context of radiosity methods, [?].

The two methods above each have their own advantages and disadvantages.

The *geometric approach* works in object space, and hence it is resolution independent. The complexity of the method however depends on the number N_p of polygons in the 3-D model, and unless advanced computational geometric methods are used, the complexity scales proportional to ${N_p}^2$. This holds both for the construction of the penumbrae shadow volumes and for their interrogation during rendering. Also, many modelling paradigms (such as free-form surfaces, patches, CSG objects with (double) curved primitives, implicit surfaces, et cetera) do't lend themselves to easy computation of polyhedral penumbrae shadow volumes. Finally, although for polygonal scenes, polyhedral penumbrae shadow volumes give exact shapes of the borders of the umbrae and penumbrae regions, they don't contain information on how the shadow density function $f_{sd} = f_{sd}(x)$ varies as a function of position $x, x \in R^3$ in the penumbra. We only know that in the umbrae, $f_{sd} = 1$ and outside the penumbra, $f_{sd} = 0$. So there is additional heuristics needed in order to define f_{sd} in between.

The *signal processing approach* has to rely on some form of sampling and reconstruction, and hence it is resolution dependent. The reconstruction part has to contain a low-pass filter, otherwise either aliasing or noisy penumbrae result. The complexity, in general, will be of the order of the number of pixels times the number of samples to take from the (area) light source. Due to the sampling, a true approximation to the shadow density function f_{sd} can be obtained.

Although both approaches have received ample attention in the literature after the initial results as cited above, penumbrae continue to be a very time consuming feature in rendering. We therefore think it appropriate to propose a new approach, which can be classified as a *functional approach*. It is similar to the geometric approach in that it constructs an explicit representation for

what we call a *penumbra volume*. The method also focuses on the shape of the umbra and penumbra regions rather than on the shadow density function within the penumbra region. But since shadow density functions are slowly varying functions, and the human visual perception system is much more sensitive to high contrasts in images, for most applications it will be easy to use a similar heuristic as in the geometric approach. In contrast with the geometric approach our method can easily cope with non-polygonal shadow casting objects, and also interrogation is much more efficient than with polyhedron-based geometric models for penumbrae.

The sequel of this paper is organised as follows. In section 2, we elaborate on the idea of the functional approach to penumbrae, and we list some assumptions on which it is based. As an example we elaborate on penumbra functions for a class of implicit surface models in section 3. We present the first results of our preliminar implementation in section 4, and we discuss the merits of the method in the concluding section.

2 Penumbra functions

2.1 Definition

A penumbra function f_{sd} is a function that returns (an approximation) to the shadow density for a given light source as a function of the position in space as a value between 0 (full illumination) and 1 (full shadow).

In the signal processing approach, this function is obtained via reconstruction based on a finite number of samples.

In the geometric approach, this function is obtained as a heuristic interpolation over a geometrically defined domain (namely, the penumbra region).

In our functional approach the value of f_{sd} also results from a heuristic interpolation (see section 3), but it is composed of a number of components, where every component is the penumbra function for one of the primitive shapes in the scene. All components can be computed independently, so the complexity for interrogating the total penumbra function is linear in the number of primitive shapes. This is much more efficient than in the geometric approach where every interrogation involves inside-outside tests against complex polyhedral objects, followed by an interpolation in a complex shaped domain. It is also more efficient than in the signal processing

approach, since we dont require multiple samples, and if f_{sd} is smooth, there can be no noise or aliasing artefacts.

2.2 Composing penumbra functions for shape primitives

The composition of penumbra functions is related to the way in which the primitive shapes in the scene are composed. We elaborate on some cases.

- If a Boolean union operator is used for the primitive shapes, we propose to take the maximum of the penumbra functions. This is correct for the shapes of the umbra and penumbra regions. Indeed: if one of the penumbra functions $f_{sd;i}$ returns 1, we are in the umbra of one object and the addition of other objects cannot decrease the shadow density[1]. Similarly, the resulting penumbra function can only be 0 if all contributing penumbra functions report 0. Taking the maximum of all penumbra functions may be incorrect for the distribution of f_{sd} values within the penumbra, but since these values result from heuristic approaximations anyway, it is a reasonable approximation and produces good results in practice.

- If the primitive shapes are so called *soft objects*, as defined in [**?**] and [**?**], we construct the resulting object by adding so called *field values*. A field value for a soft object is a smooth function $v_i = v_i(x), x \in R^3$ that returns a value between 0 and 1. The total field value is the sum of all v_i in a given point, and the surface of the resulting soft object is the isosurface $\{x \in R^3 | \sum_i v_i(x) = 0.5\}$. Similar to the addition of the field values in order to obtain the resulting field function, we add the values for the corresponding penumbra functions $f_{sd;i}$ and clamp the resulting value against 0...1.

- Instead of simply adding field values in order to get soft blends between primitive shapes, say v_1 and v_2, as in [**?**], we can also take the maximum and add a contribution in the form of a decreasing positive function of $|v_1 - v_2|$. Again we can apply the same method to the associated penumbra functions.

[1] As a consequence, we may stop evaluating penumbra functions as soon as a value 1 is encountered; this saves computational effort.

In general, our approach assumes that compound shapes are constructed in such a way that, applying the same construction operator to the penumbra functions that are associated to the indiviual shape primitives, a plausible resulting total penumbra function results. In particular, this means that e.g. a Boolean subtraction operator is not allowed.

2.3 Using penumbra functions in rendering

The penumbra function can be used both in the context of scan conversion and ray tracing. In scan conversion, the evaluation of f_{sd} takes place just before pixel shading. In the context of ray tracing, evaluation of f_{sd} saves shooting (multiple) shadow rays.

Furthermore, penumbra functions could serve in the context of rendering scenes with participating media (we have not yet elaborated on this application).

3 An example of penumbra functions: implicit surfaces

We elaborate on the implementation of penumbra functions for three types of implicit primitives.

3.1 An implicit sphere

An implicit sphere is parameterised by a centre point, p, and a radius r_p. A possible definition is

$$\{x \in R^3 | (x - p \bullet x - p) - {r_p}^2 = 0\}.$$

Here, $(\bullet)$ denotes the dot product. Let the light source be centered at position s; assume that it is a sphere with radius r_s. In order to find $f_{sd}(x)$, we proceed as follows (see also Fig. 1). We first project x onto the line $l : x = p + \lambda(p - s)$; call the projection $\bar{x}$: $\bar{x} = p + \lambda(p - s)$. Requiring that $(x - \bar{x} \bullet p - s) = 0$, we find that

$$\lambda = -\frac{(p - x \bullet p - s)}{(p - s \bullet p - s)}.$$

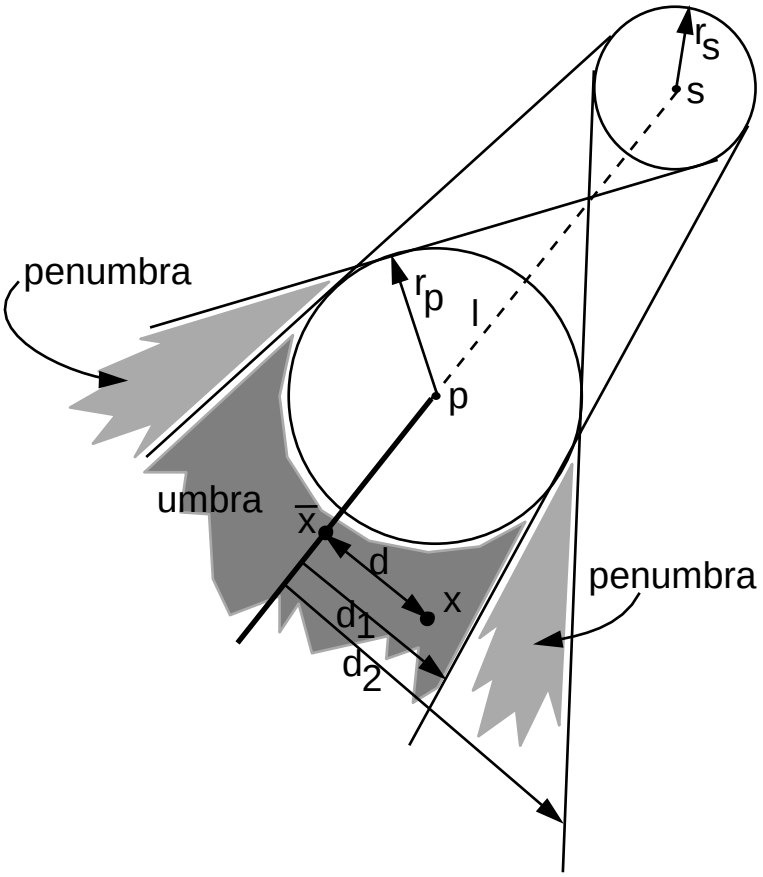

Figure 1: Figure 1: given the implicit sphere with centre p and radius r_p, and the light source with centre s and radius r_s, we can sample the penumbra function in point x. The value depends on the distance d with respect to d_1 and d_2, indicating the borders of the umbra and penumbra, respectively.

Let $d = |x - \bar{x}|$, then the value of $f_{ds}(x)$ depends solely on d. More precisely, there are two numbers, d_1 and d_2, such that

$$\begin{aligned} 0 \leq d \leq d_1 &\rightarrow f_{sd} = 1\text{: full shadow; inside umbra} \\ d_2 \leq d &\rightarrow f_{sd} = 0\text{: no shadow; outside penumbra} \\ d_1 < d < d_2 &\rightarrow f_{sd} = \frac{d_2 - d}{d_2 - d_1}\text{: outside umbra, inside penumbra} \end{aligned}$$

Here we have chosen linear interpolation as our interpolation heuristic. Of course, other interpolants could be chosen, but the only physically correct value would result from a surface integral over the visible portion of the light source. Here we do not bother to approximate the value of this integral as the value will be a slowly decreasing function of d. In our experiments, it appears that even with a linear interpolant no visible Mach banding occurs.

Also notice that d_1 could be less than 0, namely if $r_s > r_p$ and the light source is sufficiently close to the sphere. But this is physically correct: in that case the umbra region has a limited extent from behind the sphere.

We now only have to find values for d_1 and d_2. We first look at d_1. Consider the most extreme light ray that can be seen behind the sphere. This light ray is a line, say m, that is tangent both to the implicit sphere and to the light source sphere. In the 2-D plane, spanned by p, s and x, this means that we should construct the common tangent to two given circles. Now this construction is well known (see Appendix A), but it is computationally involved. If the light source is not too close to the sphere, the tangent points (=the points where m touches the circles that correspond to the sphere and the light source, respectively) can be approximated quite accurately by the points q_p and q_s, where

$$q_p = p + r_p \hat{d};$$

$$q_s = s + r_s \hat{d},$$

where $\hat{d}$ is the unit vector in the direction $x - \bar{x}$. The line m indicates the border of the umbra, so the distance between a point on m and the projection of that point onto l is d_1. Similarly, the penumbra is bounded by a line m' that touches the light source on the opposite side. Again, we approximate the true position of the tangent point by the point $q' = s - r_s \hat{d}$.

Given the lines m and m', it can be seen that both d_1 and d_2 are linear functions of λ. We know the values of d_1 and d_2 for $\lambda = 0$ and $\lambda = -1$: namely

$$\begin{aligned} \lambda = 0 &\rightarrow d_1 = r_p \text{ and } d_2 = r_s; \\ \lambda = -1 &\rightarrow d_1 = r_p \text{ and } d_2 = -r_s. \end{aligned}$$

So $d_1 = r_p + \lambda(r_p - r_s)$ and $d_2 = r_p + \lambda(r_p + r_s)$. This completes the computation for the penumbra function in the case of a sphere primitive.

3.2 An implicit line segment

Consider a line segment pq. This can be turned into an implicit line segment by assigning a radius to both ends, say r_p and r_q, respectively. If the two radii are equal, the implicit line segment will be a cylinder with hemispherical caps; otherwise, it is a cone segment capped by two hemispheres of different radii. In order to find the penumbra function for this implicit line segment, we proceed along similar lines as in section 3.1. See Fig. 2.

First, we project x onto the plane determined by p, q, and s. We call the projection $\bar{x}$. In order to achieve this, we write

$$\bar{x} = s + \lambda(p - s) + \mu(q - p).$$

Again, we have orthogonality conditions:

$$x - \bar{x} \bullet p - s) = 0$$

and

$$x - \bar{x} \bullet q - p) = 0.$$

It follows that

$$\mu = \frac{AE - BD}{CE - BF};$$

$$\lambda = \frac{AF - CD}{BF - CE},$$

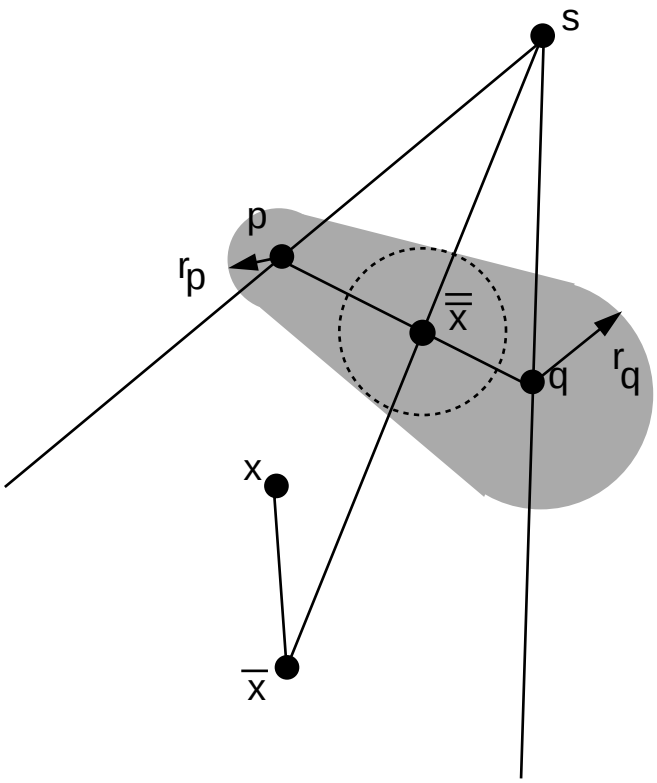

Figure 2: Figure 2: the steps in order to get the penumbra function for an implicit line segment: first, project the sample point x onto the plane through p, q, s. Next, project onto the line segment pq, Using this projection, $\bar{x}$, find the required radius in the point $\bar{\bar{x}}$. Then act as with the implicit sphere.

where

$$A = (x - s \bullet p - s);$$

$$B = (p - s \bullet p - s);$$

$$C = (q - p \bullet p - s);$$

$$D = (x - s \bullet q - p);$$

$$E = (p - s \bullet q - p);$$

$$F = (q - p \bullet q - p).$$

Next, we project the point $\bar{x}$ onto the line segment pq to find $\bar{\bar{x}}$. We write

$$\bar{\bar{x}} = p + \nu(q - p),$$

such that

$$\bar{x} = \bar{\bar{x}} + \eta(\bar{x} - s).$$

Eliminating $\bar{\bar{x}}$, we get

$$\bar{x} = p + \nu(q - p) + \eta(\lambda(p - s) + \nu(q - p)),$$

where now $\bar{x}$, λ, and mu are known and η and ν are unknown. We introduce further abbreviations

$$G = \lambda(p - s) + \mu(q - p);$$

$$H = \bar{x} - p;$$

$$J = q - p.$$

Then the equation for η and ν becomes

$$H = \nu J + \eta G.$$

We are only interested in the value for ν, and this follows by taking cross products with G:

$$\nu = \frac{(H \times G).c}{(J \times G).c},$$

where $(vector).c$ is either the x, y or z component of $vector$. For numerical stability, we choose that component for which the denominator in absolute value is the largest.

If both H and J are parallel to G, ν cannot be determined in this way. This is the case if the light source is on the line through p and q. But in this case, the penumbra is only caused by the implicit sphere, parameterised by the end point (p or q) closest to the light source.

If we find $\nu < 0$, we assume that only the point p contributes to the formation of the penumbra; similarly, if $\nu > 1$, the penumbra is assumed to be caused by q. So in these cases we can immediately resort to the computation as outlined in 3.1.

Otherwise, once we have ν, we can assign a radius to the point $\bar{\bar{x}}$, namely $r_{\bar{\bar{x}}} = \nu r_q + (1 - \nu) r_p$. We now assume that the penumbra function in x can totally be modelled as the penumbra of a sphere with radius $r_{\bar{\bar{x}}}$, so next deal with the rest of the computation in the same way as in subsection 3.1.

3.3 An implicit triangle

In the same way as a line segment pq with radii r_p and r_q could parameterise an implicit line segment, a trianle pqr with three associated radii r_p, r_q, and r_r can parameterise an implicit triangle. Without going into the vector analytic details, we proceed by computing projections. First, we project the point x onto the plane of the triangle by computing the intersection $\bar{x}$ with the plane through pqr and the line through s and x. Next we find the barycentric coordinates α, β, and γ of $\bar{x}$:

$$\bar{x} = \alpha p + \beta q + \gamma r,$$

where α, β and γ can e.g. be found by taking cross products repeatedly with p, q and r.

If all barycentric coordinates are positive, then $\bar{x}$ lies within the triangle, and we assume[2] that the penumbra function should return 1. If $\bar{x}$ falls outside pqr, we have various cases. If only α is negative, $\bar{x}$ is on the other side of qr and we assume that we only have to take the implicit line qr into account. Similar, with negative β we look at the implicit line given by pr and with negative γ it is the implicit line given by pq. Finally, if two of the barycentric coordinates are negative, we are in the penumbra region dominated by one of the three vertices: vertex p for negative γ and β; vertex q for negative α and γ, and vertex r for negative α and β.

4 Results

In color plates ..... we show some of our experimental results. We have ....

5 Discussion and further work

Although our method is inspired by physically correct geometric arguments to find penumbra areas, there are in fact some approximations that we use. Apart from the issue regarding the precise form of the decreasing f_{sd} in the penumbra area, there are two other points that require further study.

First, if a scene is composed of multiple primitives where some form of blending is used, the blend between two primitives will not cast a shadow (umbra nor penumbra) if the penumbra is too small. This is due to the fact that the blend is really an artefact from adding field values that are both smaller than the isovalue, but whose sum is larger than the isovalue. This is understood most easily in the case that $r_s = 0$. In that case, there are no penumbrae, so we only get umbrae due to all the primitives in isolation. But the shape of the surface is the result of combinations of primitives, so the blended regions cannot cast umbrae. Hence, for small but non zero values for r_s, there will be missing umbrae as well.

There is a simple partial remedy to this problem. Consider again the situation as in Fig. 1. Assume that the implicit sphere not only has a radius r_s, which is the radius for the implicit surface in the absence of any other

[2]This is not necessarily correct, namely if the light source is large compared to the size of the triangle, it is possible that there is penumbra in the truncated pyramid with apex s underneath pqr. We study this situation further in the Discussion section.

primitives (so with no blending), but also a radius r_{ss}, which is the radius for which the field contribution of this primitive decays to 0. Of course, $r_{ss} > r_s$. Now if we take the radius r_s to compute the border of the umbra (i.e., the value of d_1, and r_{ss} to compute the border of the penumbra (i.e., the value of d_2), then also the blended portion will cast (pen)umbrae. However, the penumbra area is then wider than the physically correct penumbra.

Second, if light sources are large with respect to primitives, there can be penumbrae 'underneath' these primitives. 'Underneath' here means: in regions from which the centre of the light source cannot be seen. Our method produces penumbrae in the physically correct regions. However, if two of these primitives are adjacent, in the physical situation these penumbrae become umbrae because of one primitive blocking the light rays that should illuminate the other's penumbra region. Now since our method deals with penumbra functions independently, there is no way that we can detect if another primitive is blocking the light in a penumbrae area. This problem however only occurs with very large light sources, and for a physically correct approach in these cases it is doubtful how reliable an essentially local method can be.

Finally, our method currently supports no CSG combination operators other than union. We plan to investigate if the basic concept of decomposing penumbrae by means of penumbrae functions can be extended to accommodate for intersections and differences as well.

6 Appendix: a construction for the tangent to two circles

At some places in our derivations, we had to find the tangent to two given circles. If these two circles have the same radii, this problem is trivial. Consider the case as in Fig. 3, where, without loss of generality, $r_s < r_p$. The desired tangent points iare q_s and q_p. Observe that sq_s is parallel to pq_p. Now reduce the radii of both circles with r_s.For the new radii we have $r'_p = r_p - r_s$ and $r'_s = 0$. Then for the new tangent points, we have $q'_s = s$, and we only have to find the point q'_p. Notice that by shrinking the radii of both circles with the same amount, $q'_s q'_p$ stays parallel to $q_s q_p$.The triangle

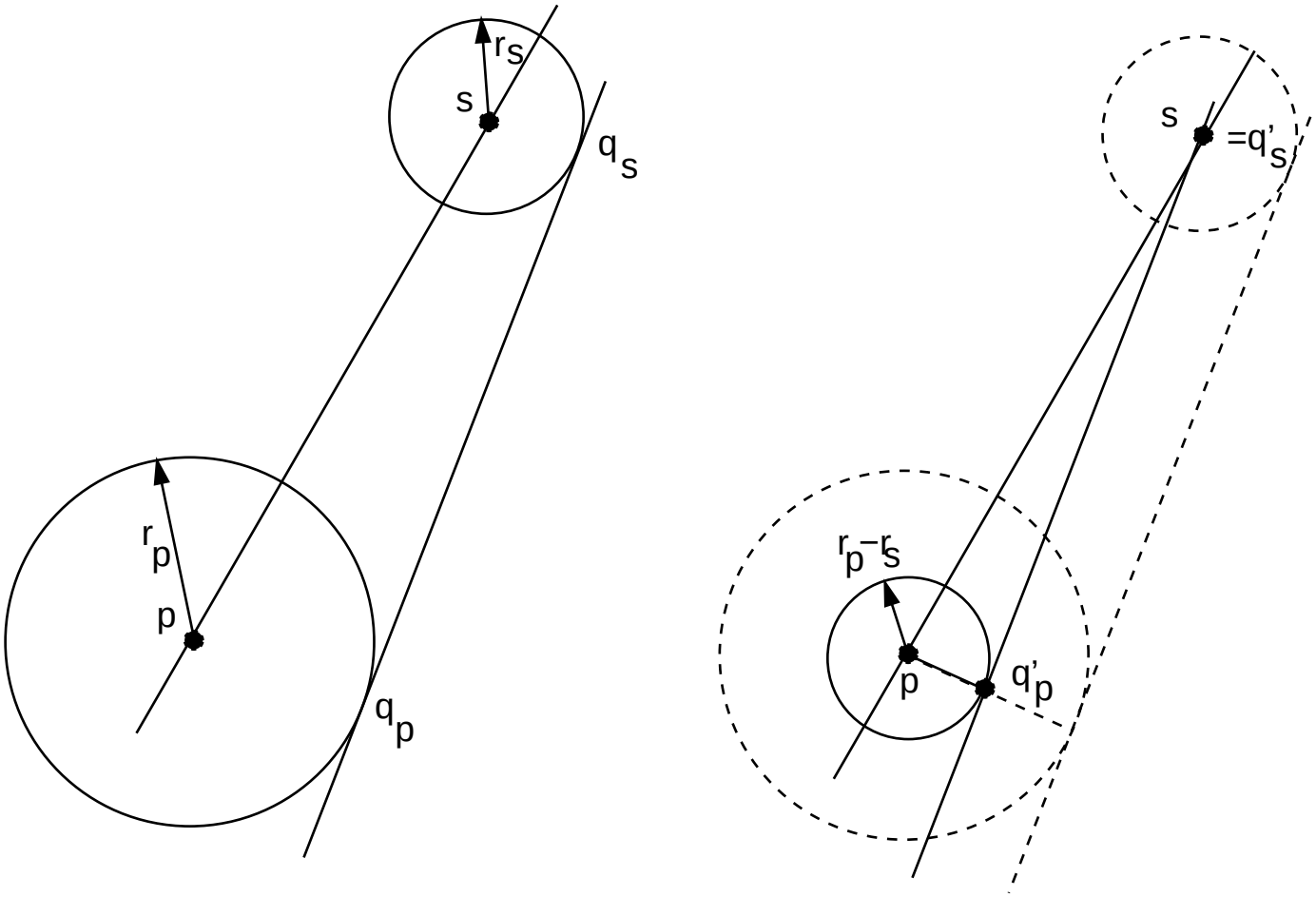

Figure 3: Figure 3: the construction of a tangent to two given circles. Left: the original situation, right: the situation after the circle with centre s has been reduced to a point circle. Since spq'_p is a rectangular triangle, the position of q'_p can easily be found.

spq'_p is rectangular, and the angle θ in point s is

$$\theta = \arcsin \frac{|p - q'_p|}{|s - p|}.$$

So we can compute the coordinates of q'_p with respect to a coordinate system which has its origin in p and the y-axis aligned with $s - p$. The slope of the line pq'_p is θ, so the y-coordinate of q'_p is $(r_p - r_s)\sin(\theta)$ and the x-coordinate is $(r_p - r_s)\cos(\theta)$. The only step that is left than is to multiply the position of q'_p with respect to p with a factor of $\frac{r_p}{r_p - r_s}$. The situation where we need the tangent that touches one of the two circles at the inside is analogous.